Termine:

Mo. 14.00 - 15.30 AH III
Mi. 14.00 - 15.30 AH II

Übungen:

Di. 14.00 - 15.30 Raum 6019
Di. 15.30 - 17.00 Raum 6019

Zuordnung:

Theoretische Informatik, Vertiefungsgebiet Algorithmen

Wichtig:

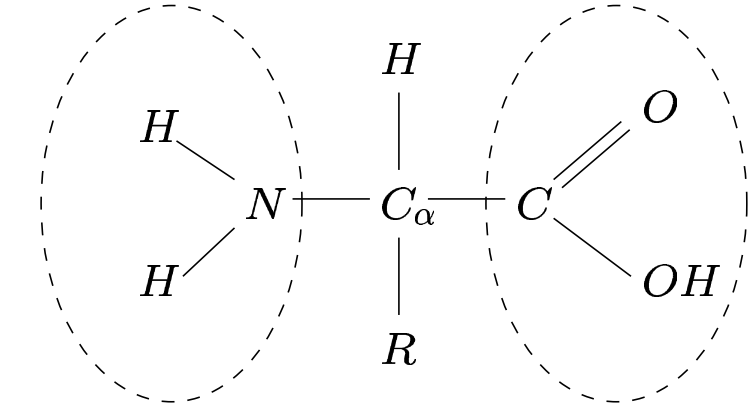
Vorlesungen am 5.5. und am 7.5. fallen aus!

Übungen:

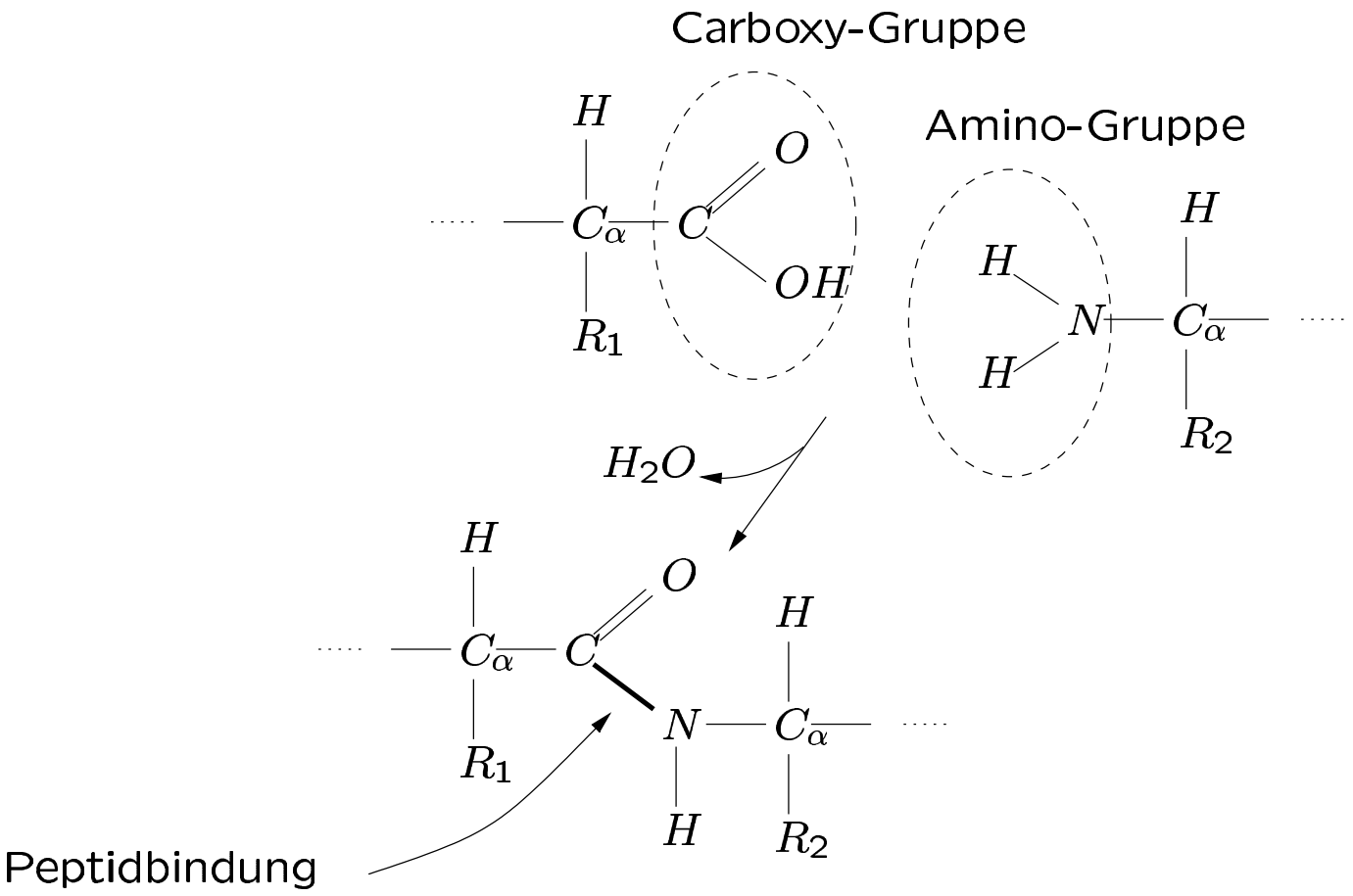
29.4. : Einführung in die Grundlagen der Algorithmik
13.5. : Beginn der regulären Übungen

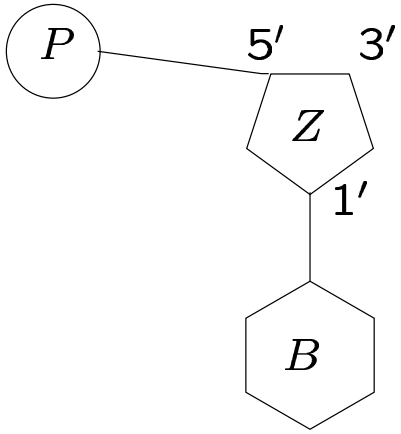
Literatur:

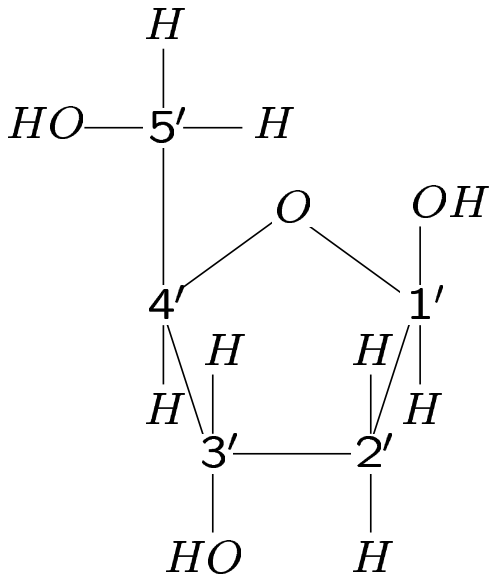
Böckenhauer, Bongartz: Algorithmische Grundlagen der Bioinformatik.
Teubner 2003 (Anfang bis Mitte Mai)


H
H
N
Cα
C
O
H
OH
R
Amino-Gruppe
Carboxy-Gruppe

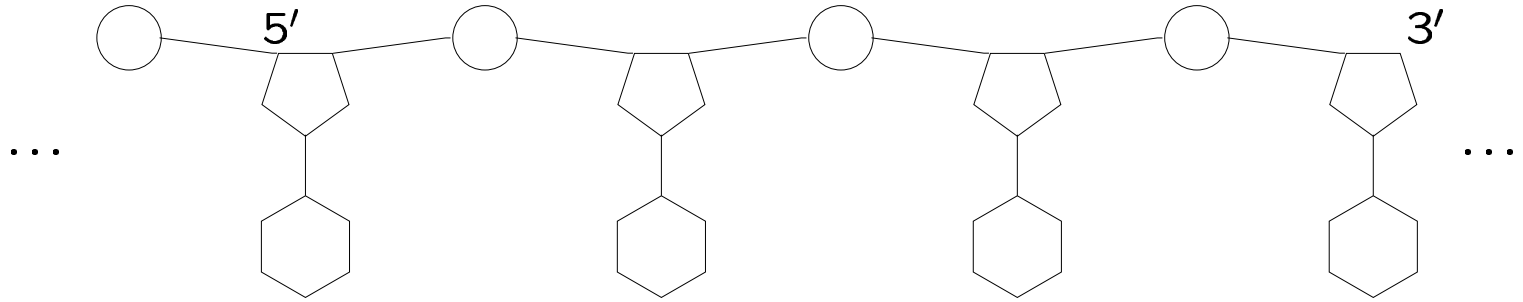
Name	Alanin	Valin	Leucin	Isoleucin	Phenylalanin
Abkürzung	Ala	Val	Leu	Ile	Phe
Code	A	V	L	I	F
Name	Prolin	Methionin	Serin	Threonin	Cystein
Abkürzung	Pro	Met	Ser	Thr	Cys
Code	P	M	S	T	C
Name	Tryptophan	Tyrosin	Asparagin	Glutamin	Asparaginsäure
Abkürzung	Trp	Tyr	Asn	Gln	Asp
Code	W	Y	N	Q	D
Name	Glutaminsäure	Lysin	Arginin	Histidin	Glycin
Abkürzung	Glu	Lys	Arg	His	Gly
Code	E	K	R	H	G

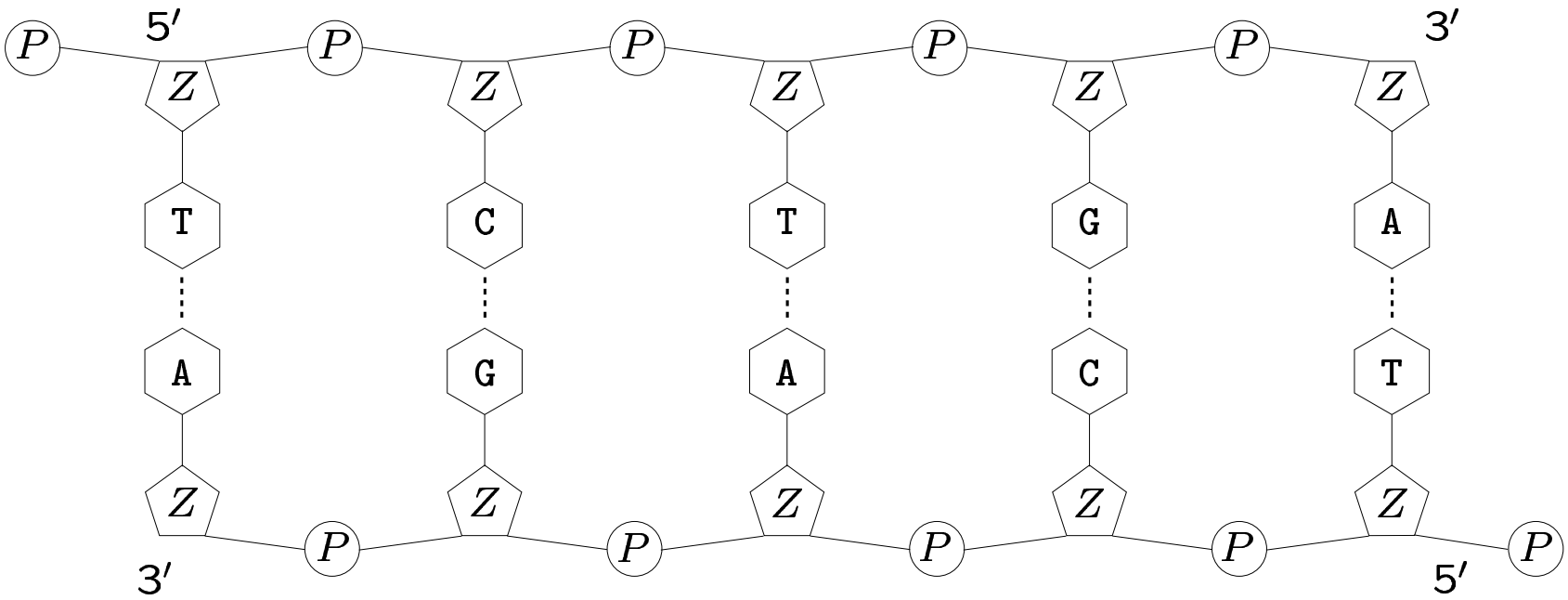

Carboxy-Gruppe
Amino-Gruppe
H_2O
Peptidbindung

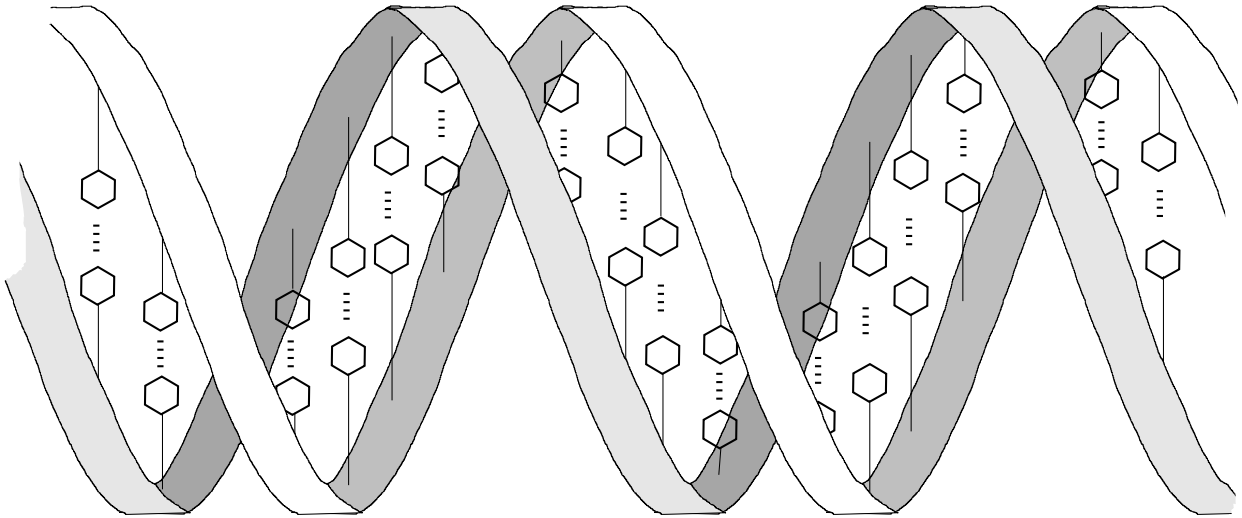


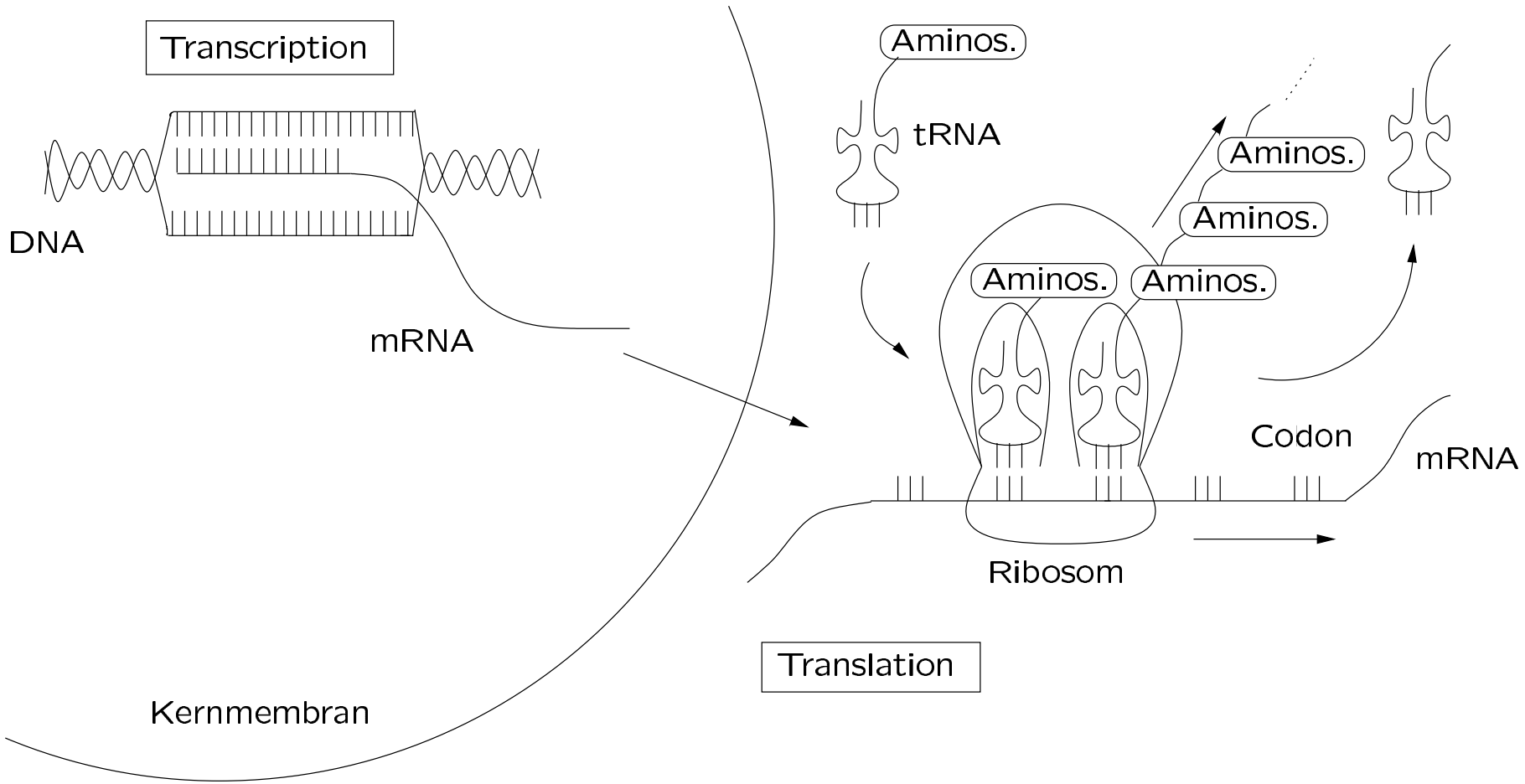


B steht für eine der Basen `A` = Adenin, `C` = Cytosin, `G` = Guanin, und `T` = Thymin (oder `U` = Uracil bei RNA).


5′
3′
...
...

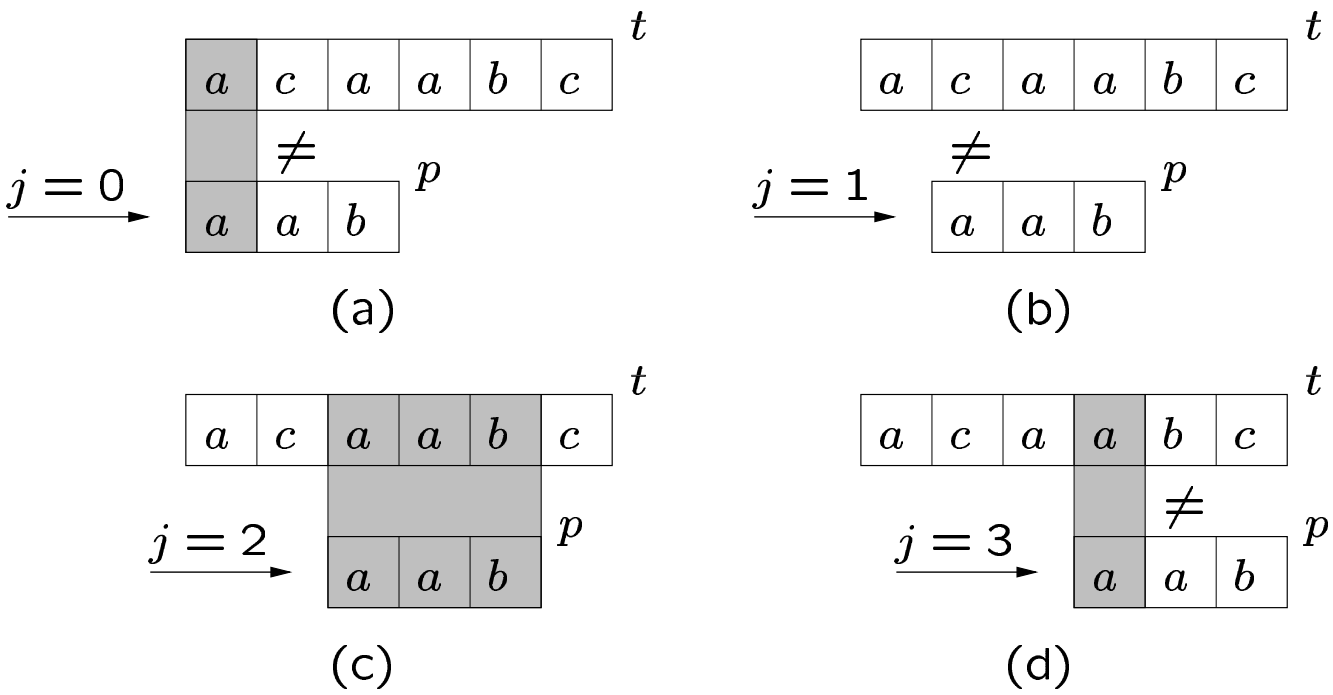

5′
3′
P
Z
T
C
T
G
A
A
G
A
C
T
3′
5′

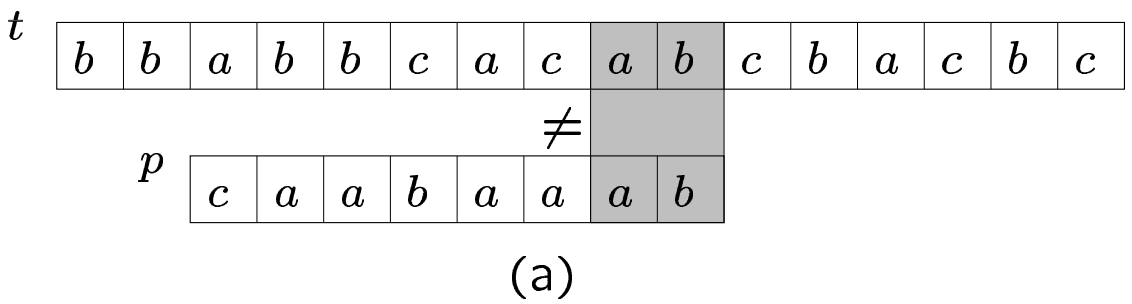



Transcription
DNA
mRNA
Kernmembran
Aminos.
tRNA
Aminos.
Aminos.
Aminos.
Aminos.
Codon
mRNA
Ribosom
Translation

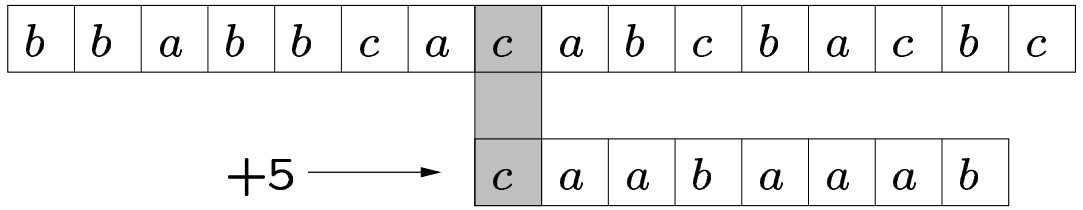
Erste Position	Zweite Position G	A	C	U	Dritte Position
G	Gly	Glu	Ala	Val	G
	Gly	Glu	Ala	Val	A
	Gly	Asp	Ala	Val	C
	Gly	Asp	Ala	Val	U
A	Arg	Lys	Thr	Met	G
	Arg	Lys	Thr	Ile	A
	Ser	Asn	Thr	Ile	C
	Ser	Asn	Thr	Ile	U
C	Arg	Gln	Pro	Leu	G
	Arg	Gln	Pro	Leu	A
	Arg	His	Pro	Leu	C
	Arg	His	Pro	Leu	U
U	Trp	STOP	Ser	Leu	G
	STOP	STOP	Ser	Leu	A
	Cys	Tyr	Ser	Phe	C
	Cys	Tyr	Ser	Phe	U

- String-Algorithmen / String-Matching
- Alignment-Verfahren
- Physikalische Kartierung
- Sequenzierung
- Signalbestimmung in DNA-Sequenzen
- Vergleich von Genomen
- Phylogenetische Bäume
- Strukturvorhersagen

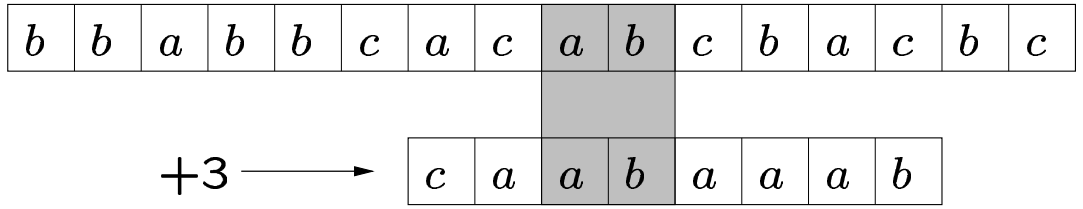

t
a c a a b c
j = 0
≠
p
a a b
(a)
t
a c a a b c
j = 1
≠
p
a a b
(b)
t
a c a a b c
j = 2
p
a a b
(c)
t
a c a a b c
j = 3
≠
p
a a b
(d)


t
b b a b b c a c a b c b a c b c
≠
p
c a a b a a a b

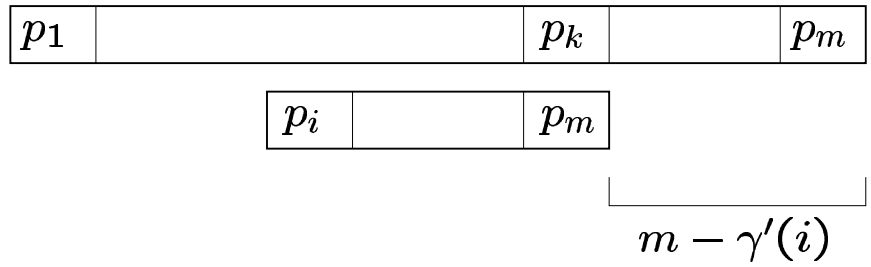
(a)


b b a b b c a c a b c b a c b c
+5
c a a b a a a b

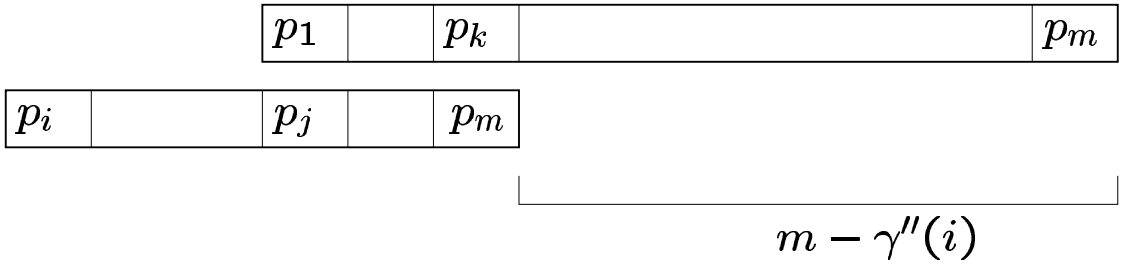
(b)


b b a b b c a c a b c b a c b c
+3
c a a b a a a b

(c)



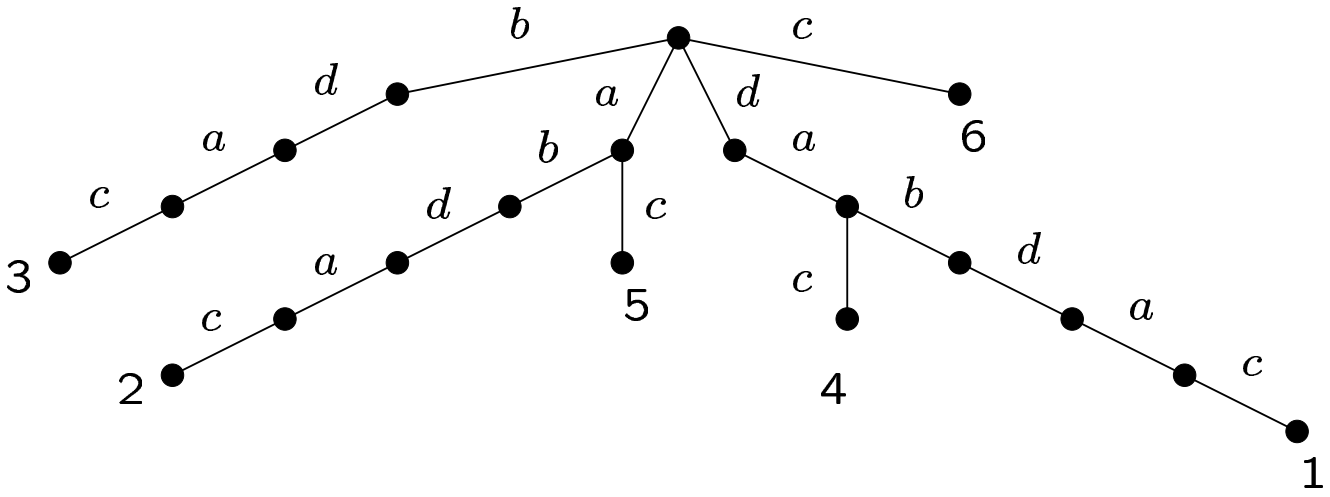
(a)



(b)

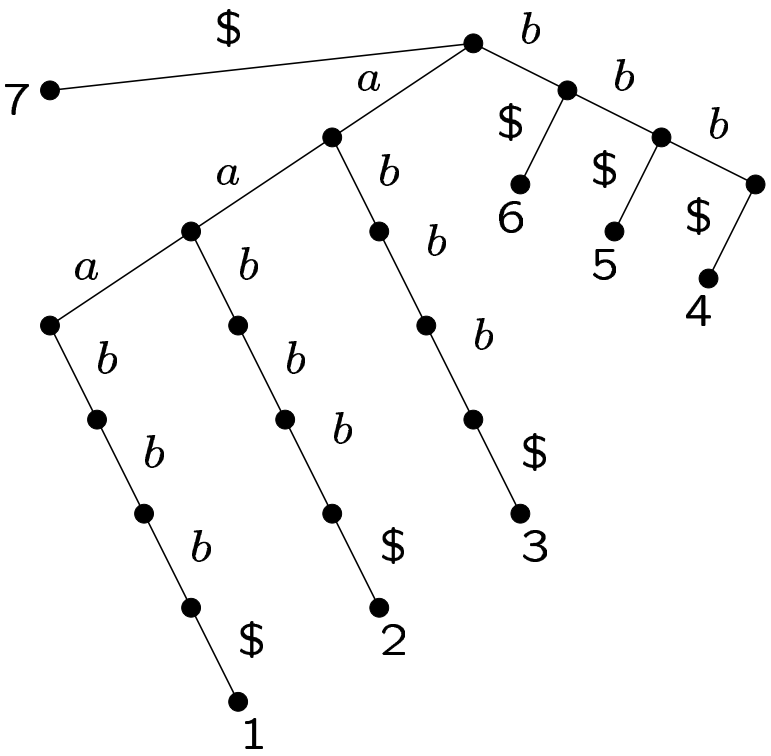
Einfacher Suffix-Baum für $t = dabdac$

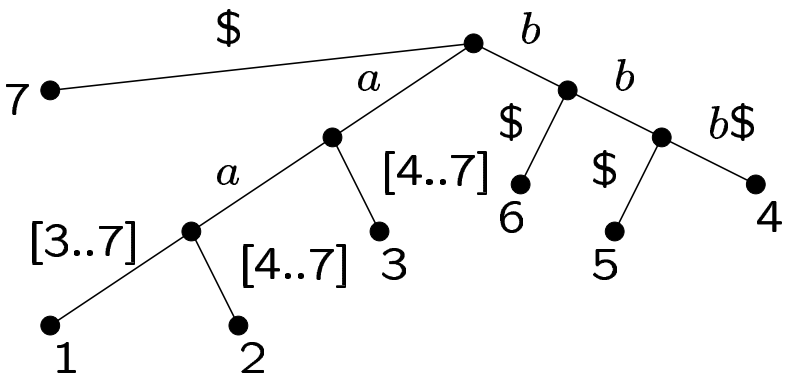
4

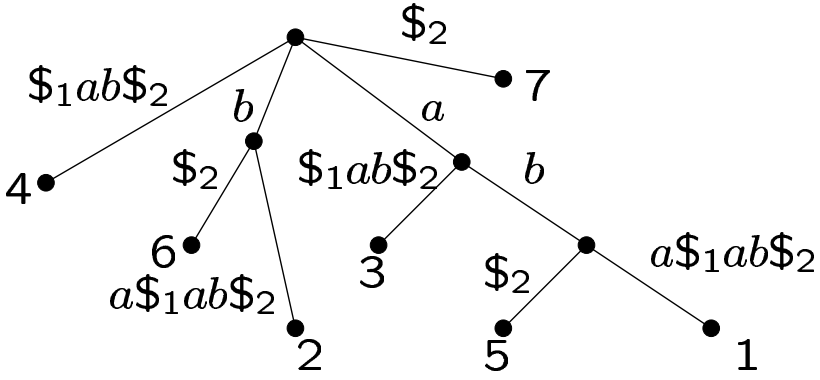


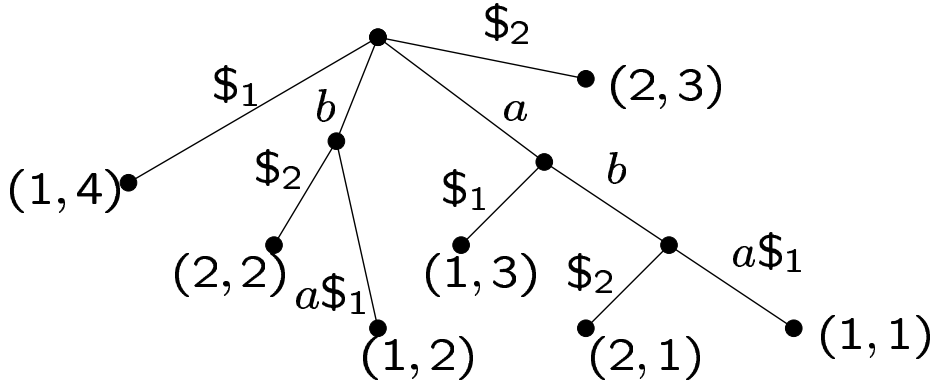
Einfacher und kompakter Suffix-Baum für $aaabbb\$$

5

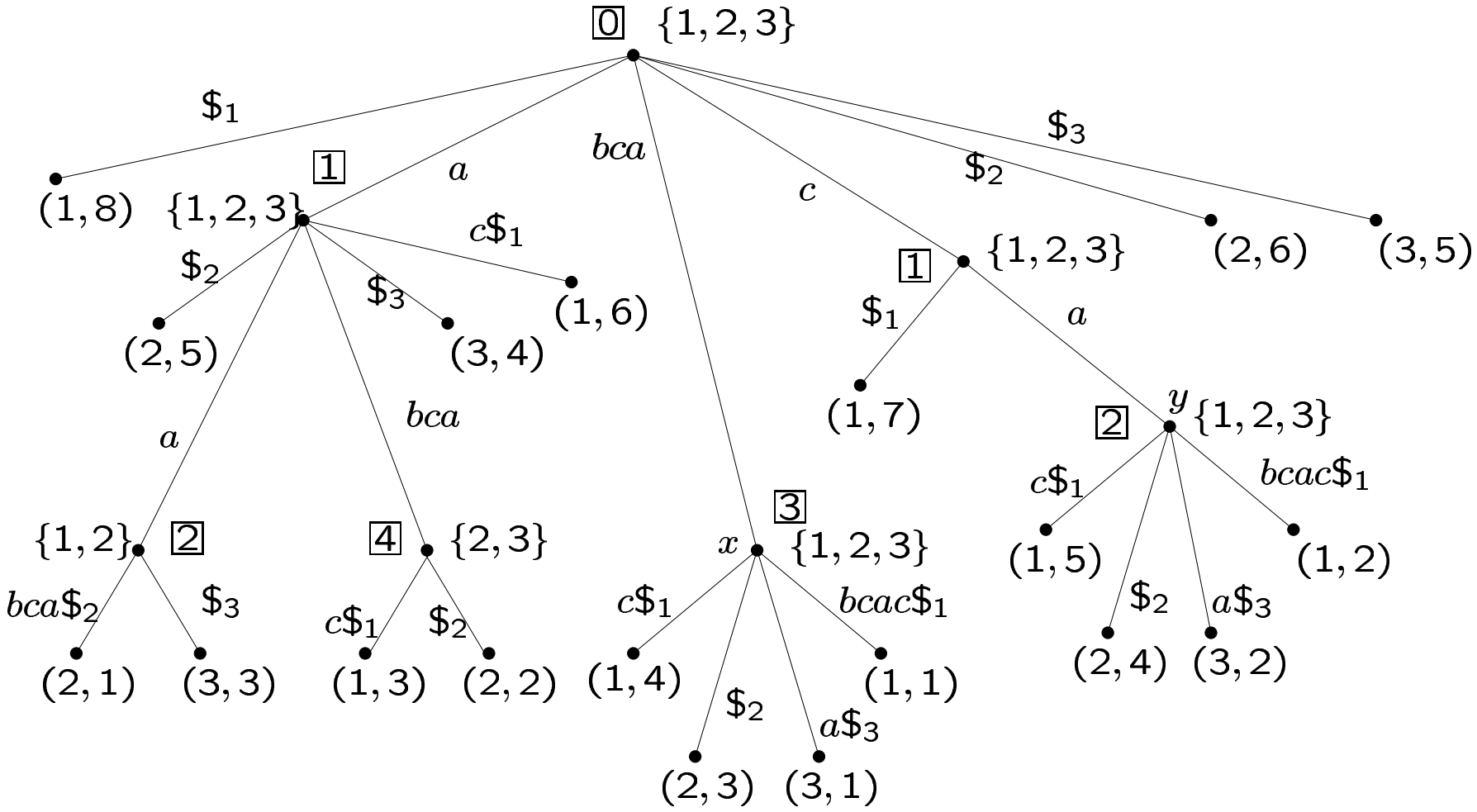




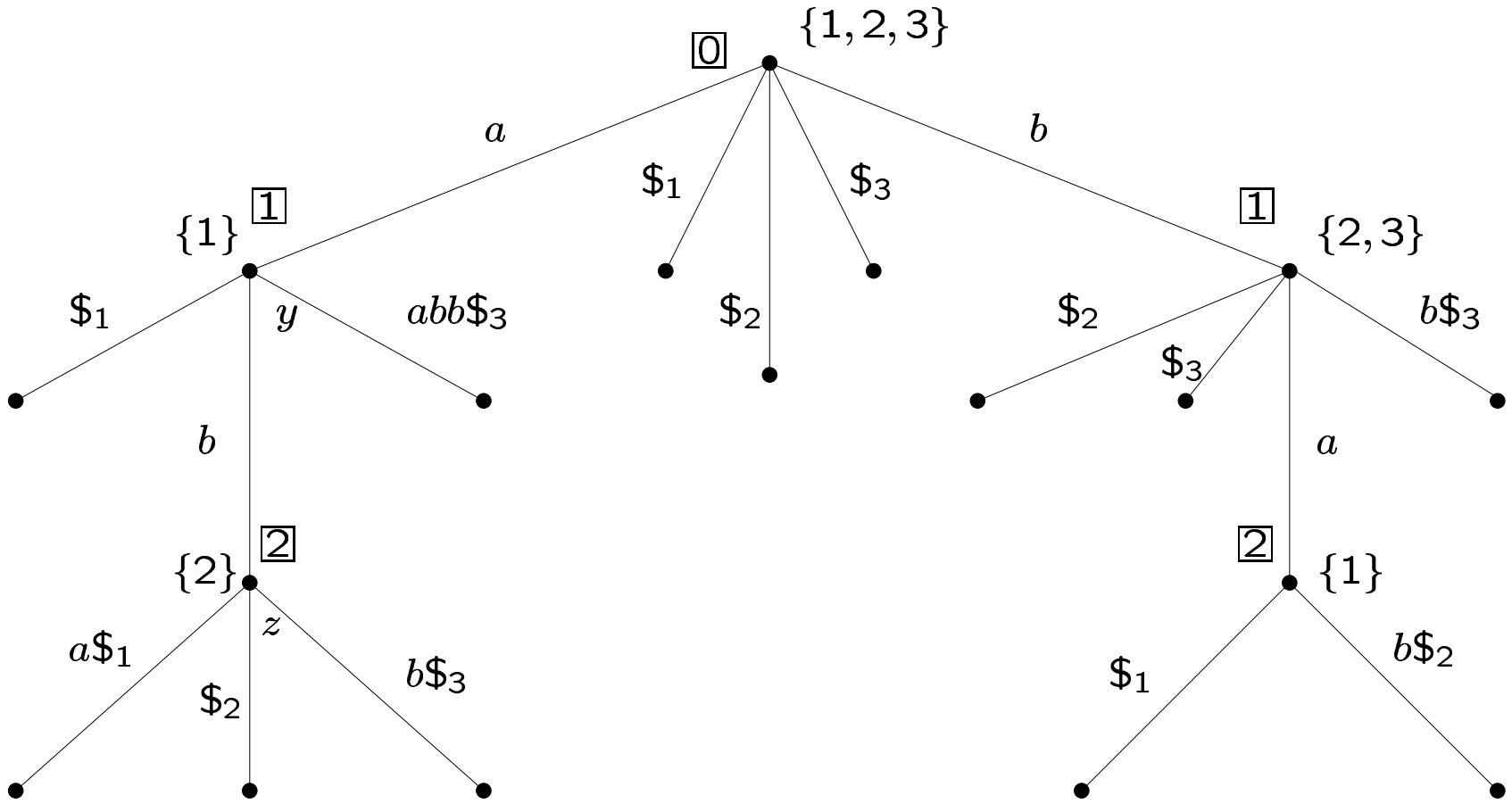

$\$_2$
$\$_1ab\$_2$
7
b
a
4
$\$_2$
$\$_1ab\$_2$
b
6
3
$\$_2$
$a\$_1ab\$_2$
$a\$_1ab\$_2$
2
5
1


$\$_2$
$\$_1$
(2, 3)
b
a
(1, 4)
$\$_2$
$\$_1$
b
(2, 2)
$a\$_1$
(1, 3)
$\$_2$
$a\$_1$
(1, 2)
(2, 1)
(1, 1)

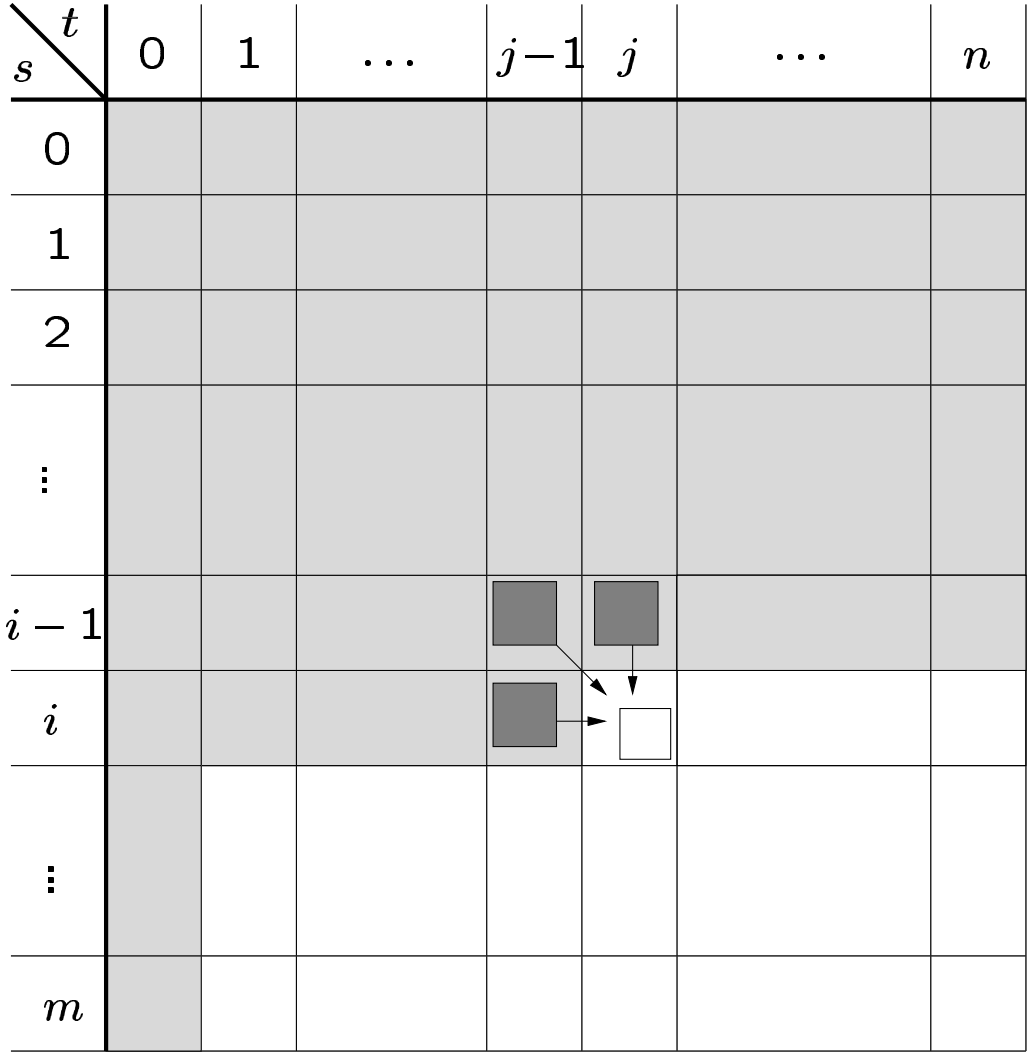
$t_1 = bcabcac$, $t_2 = aabca$, $t_3 = bcaa$



$t_1 = aba$, $t_2 = bab$, $t_3 = aabb$

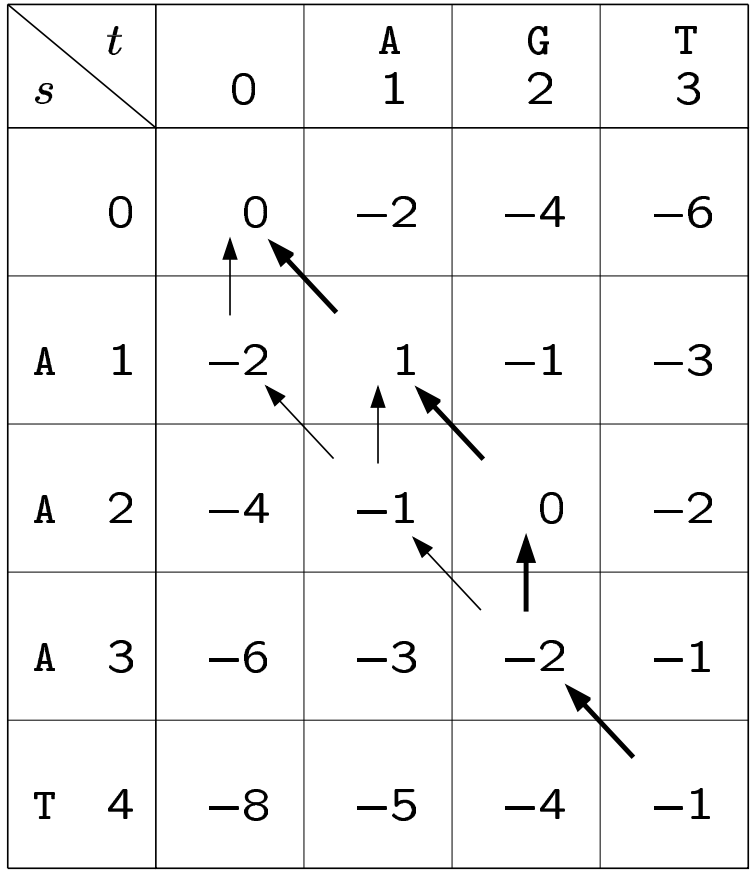


Berechnung eines globalen Alignment mittels dynamischer Programmierung



Ähnlichkeitsmatrix für die Strings $s = \texttt{AAAT}$ und $t = \texttt{AGT}$

s \ t		0	A 1	G 2	T 3
	0	0	−2	−4	−6
A	1	−2	1	−1	−3
A	2	−4	−1	0	−2
A	3	−6	−3	−2	−1
T	4	−8	−5	−4	−1



Algorithmus 5.2: Bestimmung eines optimalen Alignments

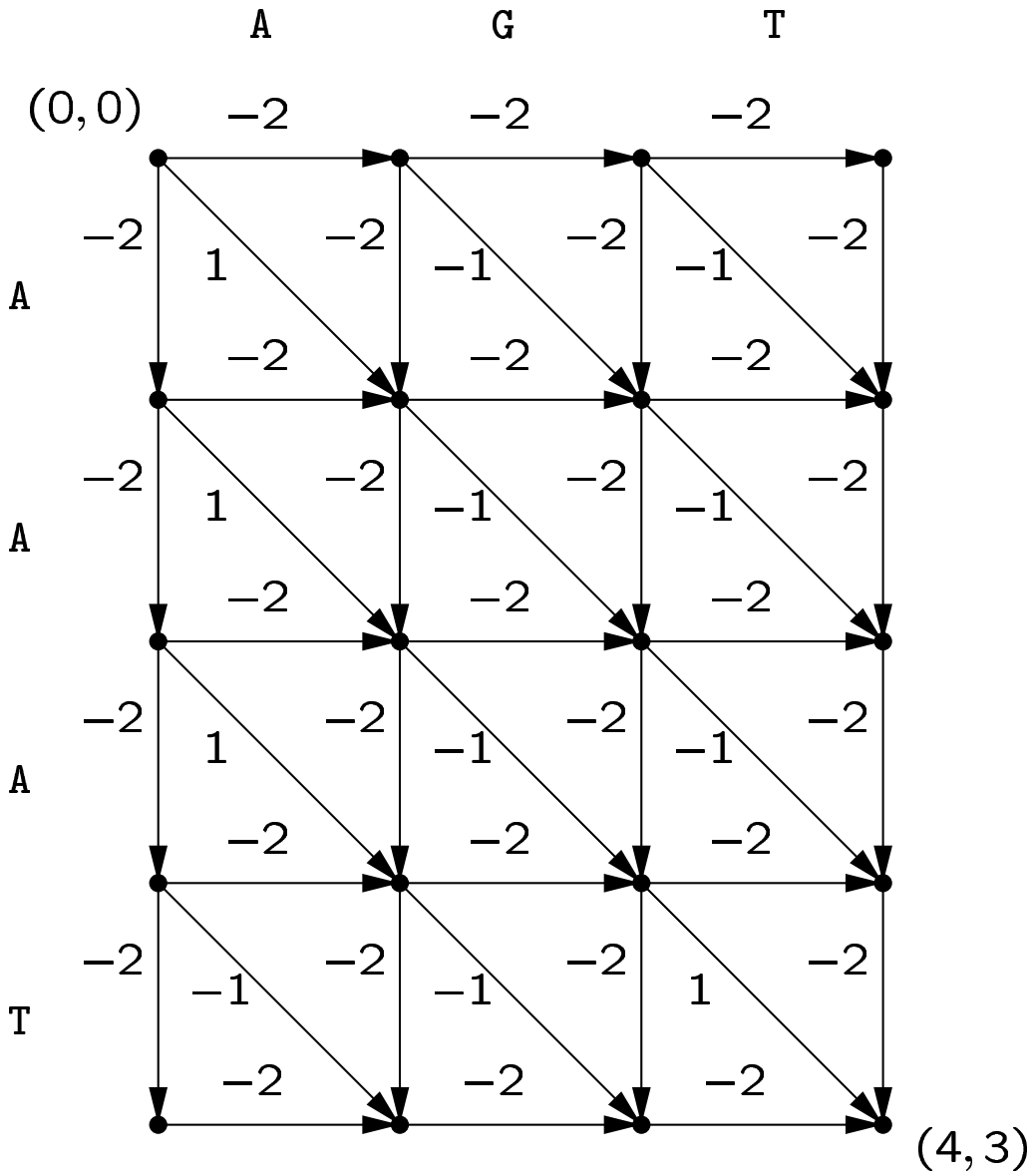
Eingabe: Eine Ähnlichkeitsmatrix M für zwei Strings $s = s_1 \dots s_m$ und $t = t_1 \dots t_n$.

Rufe die rekursive Prozedur Align(m, n) auf.

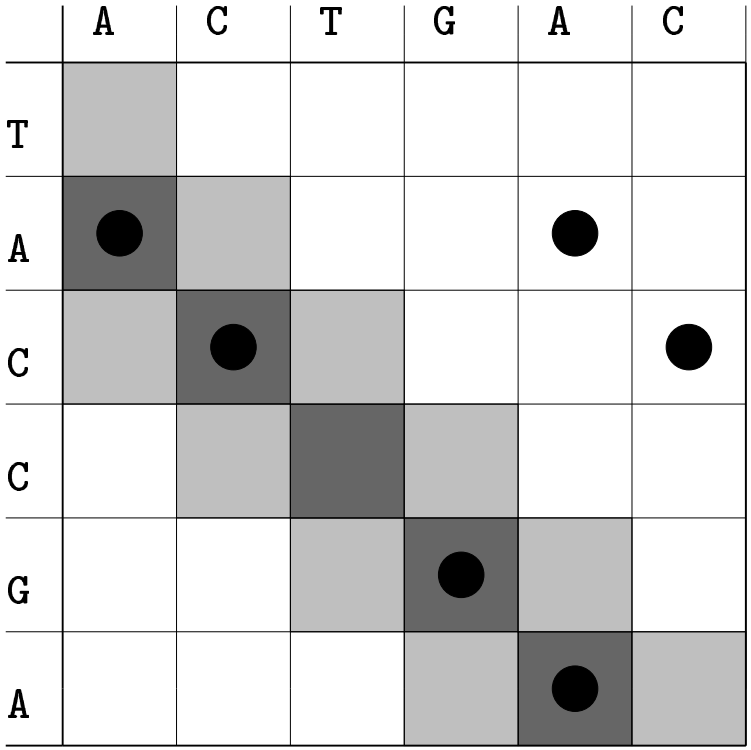
Ausgabe: Das Alignment (s', t') von s und t.

Prozedur Align(i, j)**:**

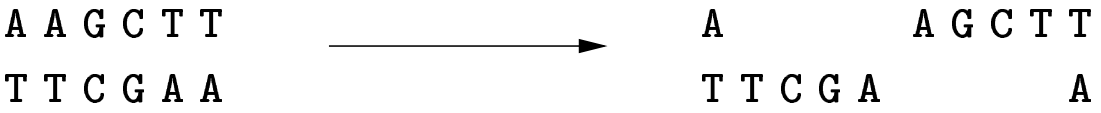
if $i = 0$ und $j = 0$ **then**
$s' := \lambda$; $t' := \lambda$
else
if $M(i,j) = M(i-1,j) + g$ **then**
$(\overline{s}, \overline{t})$:= Align$(i-1, j)$
$s' := \overline{s} \cdot s_i$; $t' := \overline{t} \cdot$ -
else if $M(i,j) = M(i,j-1) + g$ **then**
$(\overline{s}, \overline{t})$:= Align$(i, j-1)$
$s' := \overline{s} \cdot$ - ; $t' := \overline{t} \cdot t_j$
else $\{M(i,j) = M(i-1,j-1) + p(s_i, t_j)\}$
$(\overline{s}, \overline{t})$:= Align$(i-1, j-1)$
$s' := \overline{s} \cdot s_i$; $t' := \overline{t} \cdot t_j$
return (s', t')


A
G
T
(0,0)
A
A
A
T
(4,3)

s \ t		0	A 1	G 2	T 3	A 4
	0	0	−2	−4	−6	−8
A	1	0	1	−1	−3	−5
A	2	0	1	0	−2	−2
A	3	0	1	0	−1	−1
T	4	0	−1	0	1	−1


A C T G A C
T
A
C
C
G
A

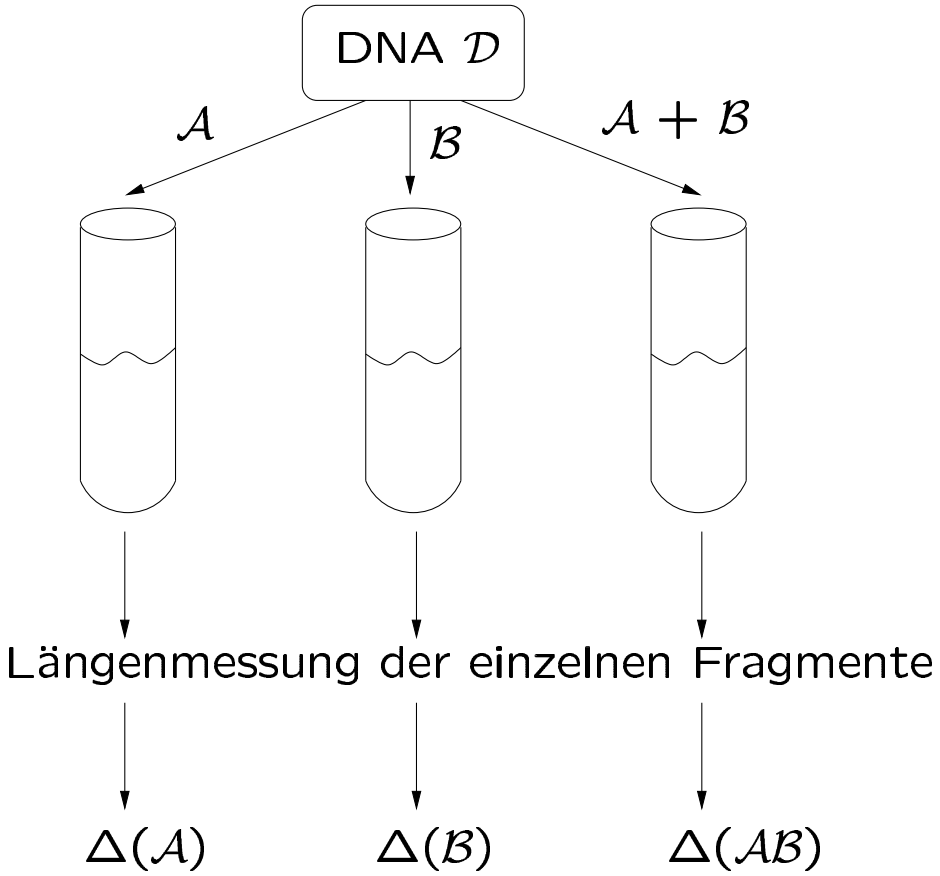
δ	0	1	a	b	$-$
0	2	2	1	2	1
1	2	2	2	1	1
a	1	2	0	k	1
b	2	1	k	0	1
$-$	1	1	1	1	0


A A G C T T
T T C G A A
A A G C T T
T T C G A A

Eingabe: DNA-Molekül $\mathcal{D}$ und zwei Restriktionsenzyme $\mathcal{A}$ und $\mathcal{B}$.

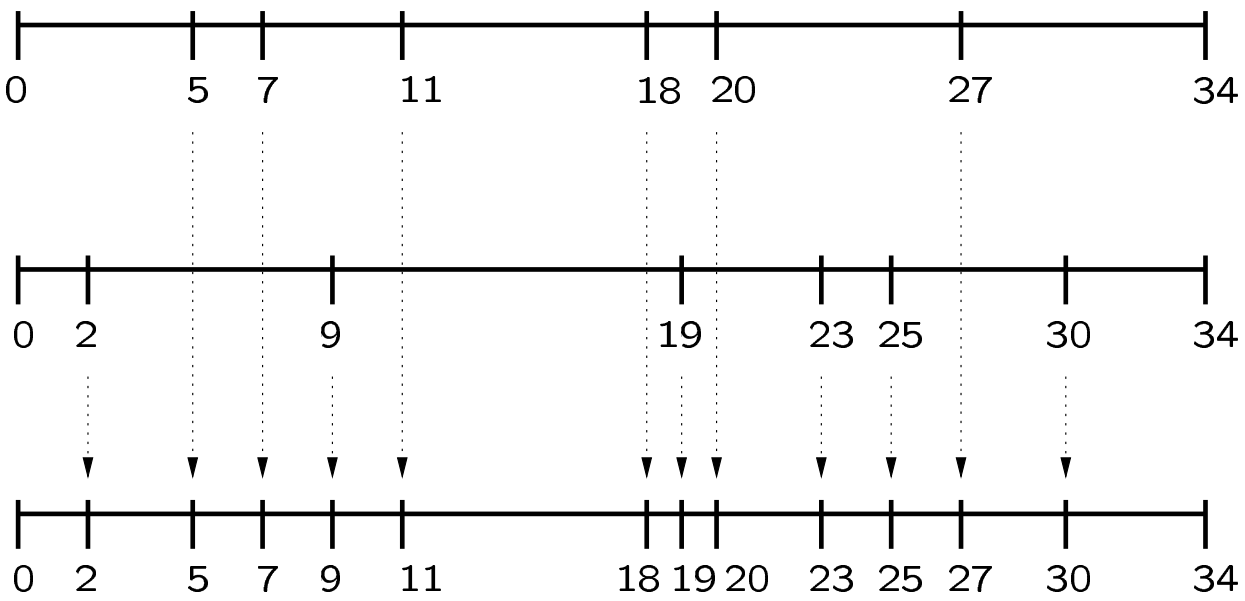
1. Erzeuge drei Kopien von $\mathcal{D}$.
2. Verdaue erste Kopie mit Enzym $\mathcal{A}$, zweite Kopie mit $\mathcal{B}$ und dritte Kopie mit Enzym $\mathcal{A}$ und $\mathcal{B}$ zusammen.
3. Bestimme die Längen dieser so erhaltenen Fragmente von $\mathcal{D}$. Es ergeben sich die Multimengen:
   - $\Delta(\mathcal{A})$: Enthält die Längen der Fragmente der von $\mathcal{A}$ verdauten (ersten) Kopie.
   - $\Delta(\mathcal{B})$: Enthält die Längen der Fragmente der von $\mathcal{B}$ verdauten (zweiten) Kopie.
   - $\Delta(\mathcal{AB})$: Enthält die Längen der Fragmente der von $\mathcal{A}$ und $\mathcal{B}$ verdauten (dritten).

Ausgabe: Die Multimengen $\Delta(\mathcal{A})$, $\Delta(\mathcal{B})$, $\Delta(\mathcal{AB})$.


DNA $\mathcal{D}$
$\mathcal{A}$
$\mathcal{B}$
$\mathcal{A}+\mathcal{B}$
Längenmessung der einzelnen Fragmente
$\Delta(\mathcal{A})$
$\Delta(\mathcal{B})$
$\Delta(\mathcal{AB})$

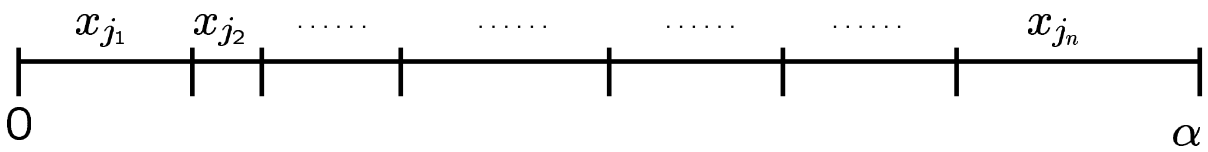
$A = \{2,2,4,5,7,7,7\}$
$\pi = (5,2,4,7,2,7,7)$

$B = \{2,2,4,4,5,7,10\}$
$\phi = (2,7,10,4,2,5,4)$

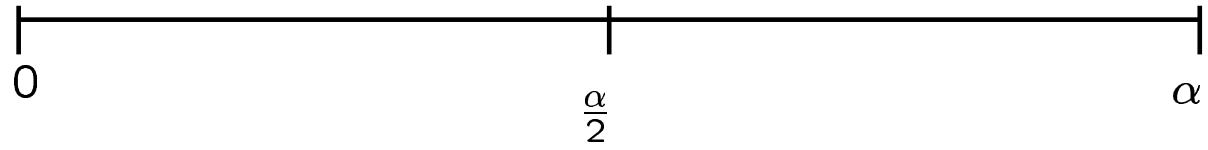


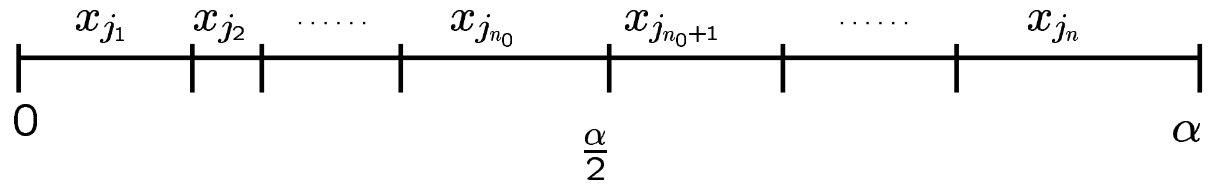
$C = \{1,1,2,2,2,2,2,2,3,3,3,4,7\}$
$Pos(\pi) \cup Pos(\phi) = \{0,2,5,7,9,11,18,19,20,23,25,27,30,34\}$

$A = X = \{x_1, \ldots, x_n\}$



$B = \{\frac{\alpha}{2}, \frac{\alpha}{2}\}$





$C = X$

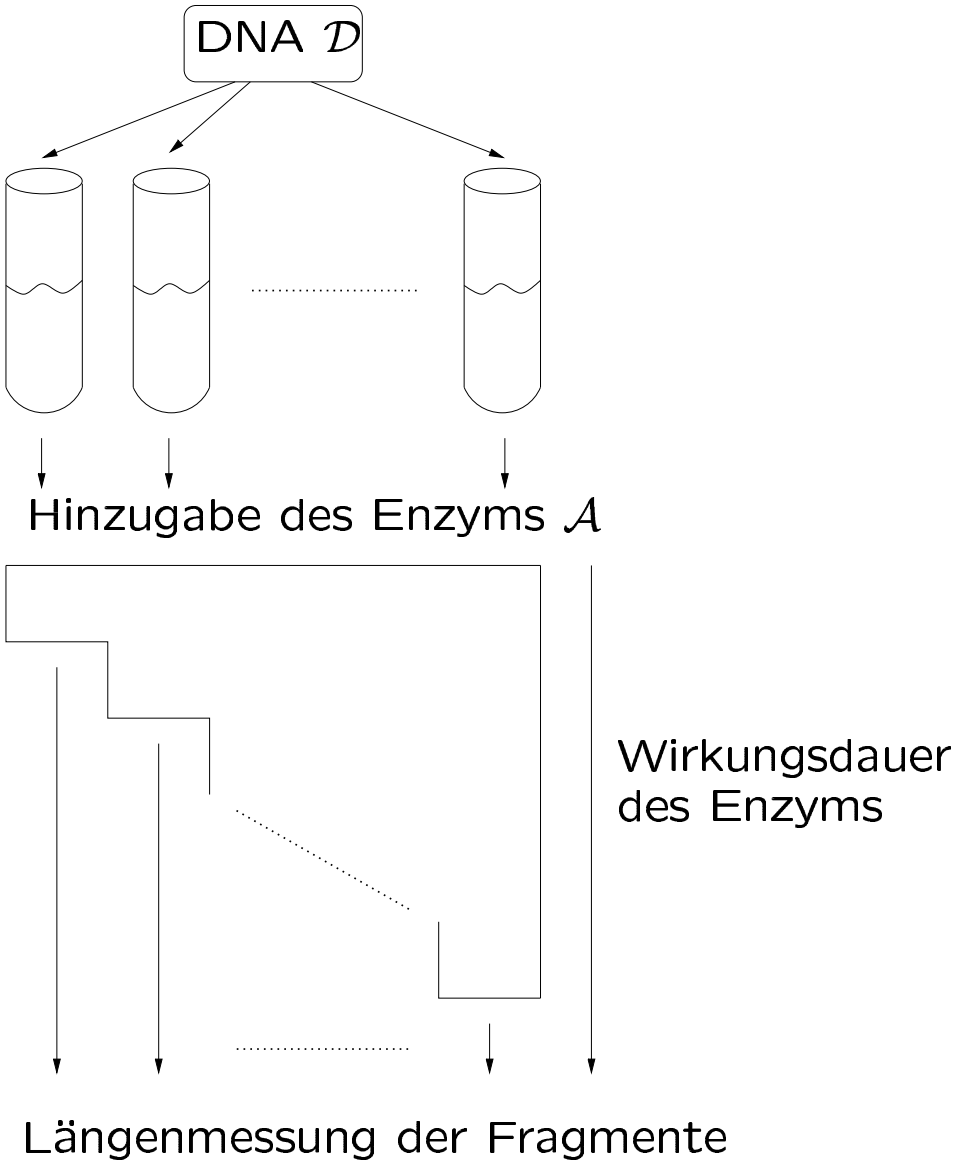
Partitionen: $Y = \{x_{j_1}, \ldots, x_{j_{n_0}}\}$

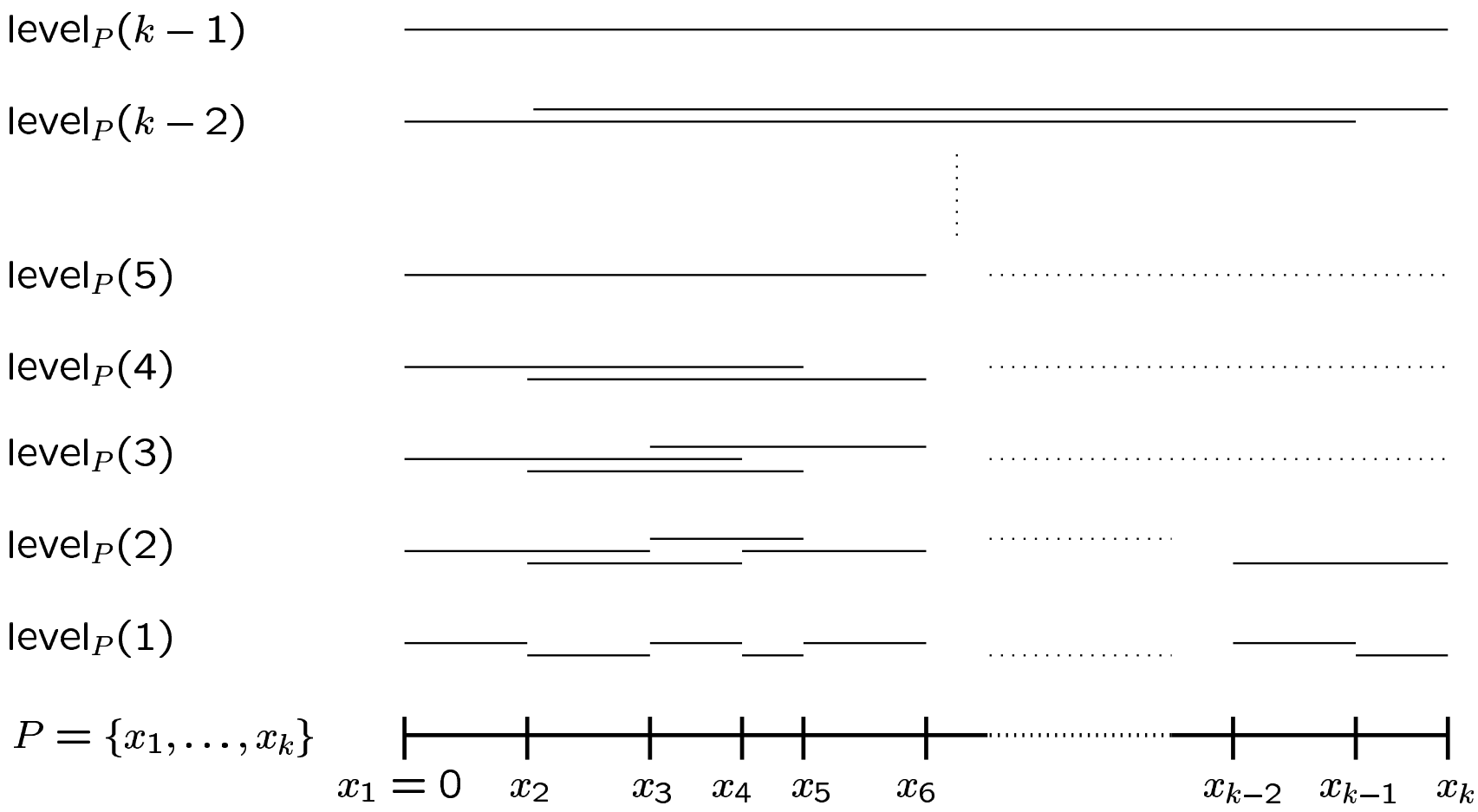
$Z = \{x_{j_{n_0+1}}, \ldots, x_{j_n}\}$

Eingabe: DNA-Sequenz $\mathcal{D}$ und ein Restriktionsenzym $\mathcal{A}$.

1. Erzeuge mehrere Kopien von $\mathcal{D}$.
2. Verdaue in getrennten Ansätzen die Kopien durch Enzym $\mathcal{A}$ mit unterschiedlicher Wirkungsdauer.
3. Bestimme die Länge der so erhaltenen Fragmente von $\mathcal{D}$ und fasse diese Längen aus allen Ansätzen in der Multimenge $\Delta_p(\mathcal{A})$ zusammen.

Ausgabe: Die Multimenge $\Delta_p(\mathcal{A})$.


DNA $\mathcal{D}$
Hinzugabe des Enzyms $\mathcal{A}$
Wirkungsdauer
des Enzyms
Längenmessung der Fragmente


level$_P(k-1)$
level$_P(k-2)$
level$_P(5)$
level$_P(4)$
level$_P(3)$
level$_P(2)$
level$_P(1)$
$P = \{x_1, \ldots, x_k\}$
$x_1 = 0$
x_2
x_3
x_4
x_5
x_6
x_{k-2}
x_{k-1}
x_k

Algorithmus 7.1:

Eingabe: Eine Multimenge A mit $\binom{k}{2}$ Elementen aus $\mathbb{N} - \{0\}$.

1. Sortiere die Elemente in A.

2. $X := \emptyset$ {bisher bestimmte Punktmenge}

3. $S :=$ leerer Stack {Stack zur Rekonstruktion der Vorgängersituation beim Backtracking}

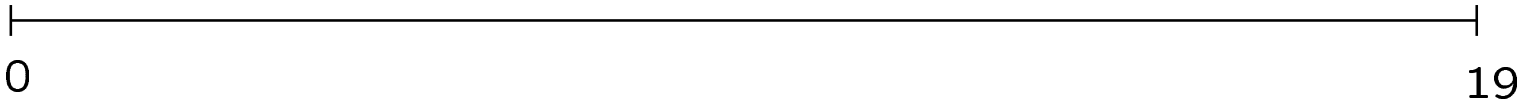
4. {Platziere das größte Fragment}
   $y_{\max} = \max A$ {Bestimme das größte Element in A.}
   $X := X \cup \{0, y_{\max}\}$ $\{x_1 := 0,\ x_k := y_{\max}\}$
   $A := A - \{y_{\max}\}$

5. Platziere die weiteren Fragmente (rechts oder links) durch einen Aufruf der rekursiven Prozedur Platziere(X, A, S).

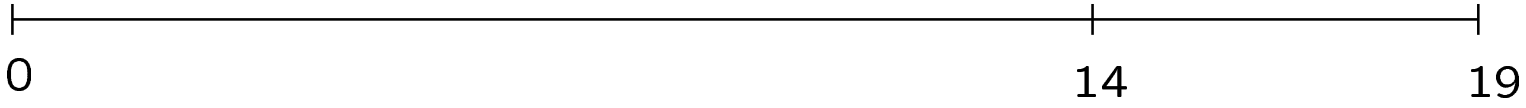
Prozedur Platziere(X, A, S)

if $A = \emptyset$ **then**
  Ausgabe: „zulässige Lösung“ X; **halt**
$y = \max A$ {Bestimme das größte Element in A.}
if $\delta(y, X) \subseteq A$ **then** {Platzierung links}
  $A := A - \delta(y, X)$
  $X := X \cup \{y\}$
  Push(y, S)
  Platziere(X, A, S)
else if $\delta(y_{\max} - y, X) \subseteq A$ **then** {Platzierung rechts}
  $A := A - \delta(y_{\max} - y, X)$
  $X := X \cup \{y_{\max} - y\}$
  Push$(y_{\max} - y, S)$
  Platziere(X, A, S)
else if $S \neq$ leerer Stack **then** {Backtracking möglich}
  $y' :=$ Pop(S) {letzte neue Position vom Stack holen}
  $A := A \cup (\delta(y', X) - \{0\})$ {alte Distanzmenge rekonstruieren}
  $X := X - \{y'\}$ {alte Positionsmenge rekonstruieren}
  return {Rücksprung in die aufrufende Prozedur}
else
  $S =$ leerer Stack {kein Backtracking möglich}
  Ausgabe: „Es existiert keine Lösung!“; **halt**

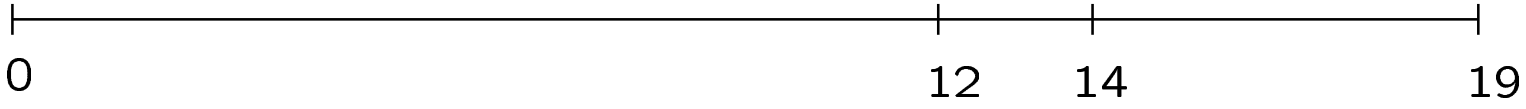
Sei $A = \{1, 2, 3, 4, 5, 5, 7, 7, 9, 9, 10, 10, 12, 14, 19\}$
Nach Ausführung der Schritte 1-4 erhalten wir:



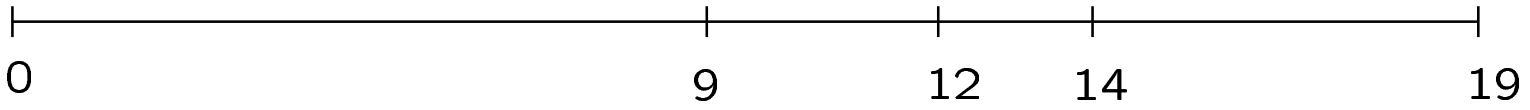
(a) $X = \{0, 19\}$; $A = \{1, 2, 3, 4, 5, 5, 7, 7, 9, 9, 10, 10, 12, 14\}$; $y = 14$; $S = ()$;
$\delta(14, X) = \{5, 14\} \subseteq A$; $\Rightarrow$ Platzierung: *links*



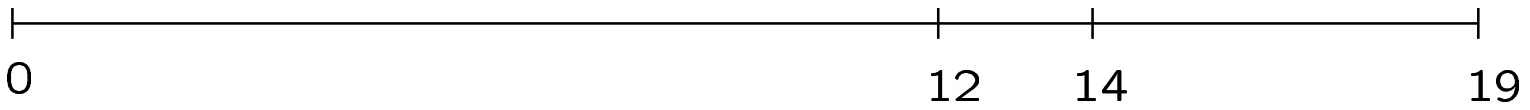
(b) $X = \{0, 14, 19\}$; $A = \{1, 2, 3, 4, 5, 7, 7, 9, 9, 10, 10, 12\}$; $y = 12$; $S = (14)$;
$\delta(12, X) = \{2, 7, 12\} \subseteq A$; $\Rightarrow$ Platzierung: *links*



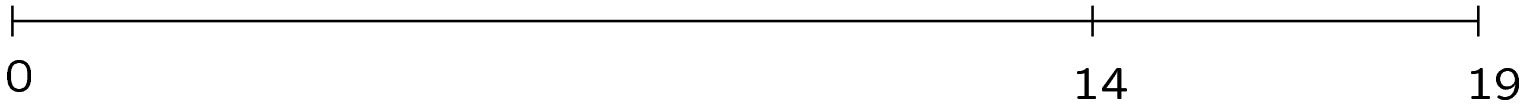
(c) $X = \{0, 12, 14, 19\}$; $A = \{1, 3, 4, 5, 7, 9, 9, 10, 10\}$; $y = 10$; $S = (14, 12)$;
$\delta(10, X) = \{2, 4, 9, 10\} \not\subseteq A$;
$\delta(19 - 10, X) = \delta(9, X) = \{3, 5, 9, 10\} \subseteq A$; $\Rightarrow$ Platzierung: *rechts*



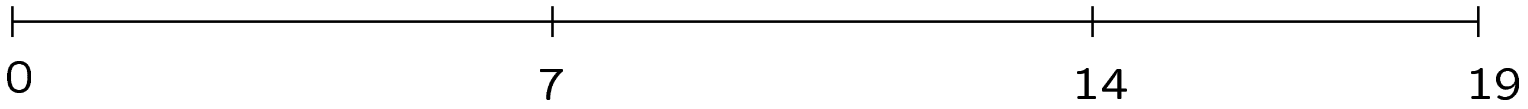
(d) $X = \{0, 9, 12, 14, 19\}$; $A = \{1, 4, 7, 9, 10\}$; $y = 10$; $S = (14, 12, 9)$;
$\delta(10, X) = \{1, 2, 4, 9, 10\} \not\subseteq A$;
$\delta(19 - 10, X) = \delta(9, X) = \{0, 3, 5, 9, 10\} \not\subseteq A$; $\Rightarrow$ *Backtracking*. Platzierung (c) wird zurückgenommen.



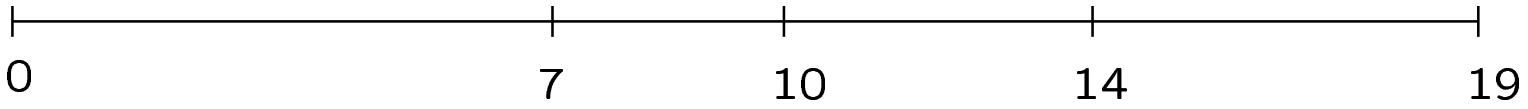
(e) $X = \{0, 12, 14, 19\}$; $A = \{1, 3, 4, 5, 7, 9, 9, 10, 10\}$; $y = 10$; $S = (14, 12)$;
$\delta(10, X) = \{2, 4, 9, 10\} \not\subseteq A$;
$\delta(19-10, X) = \delta(9, X) = \{3, 5, 9, 10\} \subseteq A$; Rückgängig gemacht im Backtracking-Schritt (d). $\Rightarrow$ *Backtracking*. Platzierung (b) wird zurückgenommen.



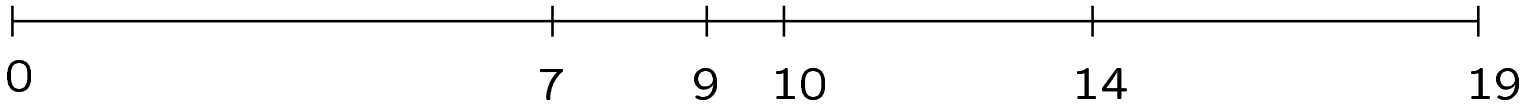
(f) $X = \{0, 14, 19\}$; $A = \{1, 2, 3, 4, 5, 7, 7, 9, 9, 10, 10, 12\}$; $y = 12$; $S = (14)$;
$\delta(12, X) = \{2, 7, 12\} \subseteq A$; Rückgängig gemacht im Backtracking-Schritt (e).
$\delta(19 - 12, X) = \delta(7, X) = \{7, 7, 12\} \subseteq A$; $\Rightarrow$ Platzierung: *rechts*



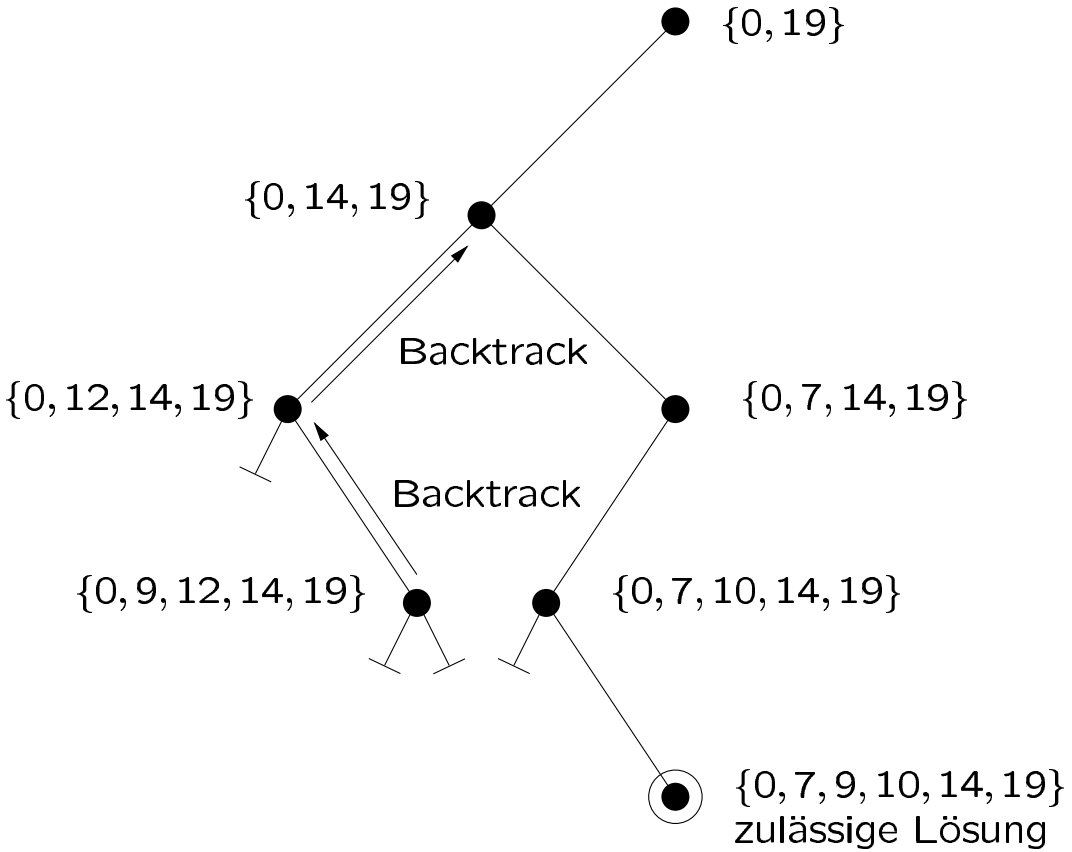
(g) $X = \{0, 7, 14, 19\}$; $A = \{1, 2, 3, 4, 5, 9, 9, 10, 10\}$; $y = 10$; $S = (14, 7)$;
$\delta(10, X) = \{3, 4, 9, 10\} \subseteq A$;
$\Rightarrow$ Platzierung: *links*

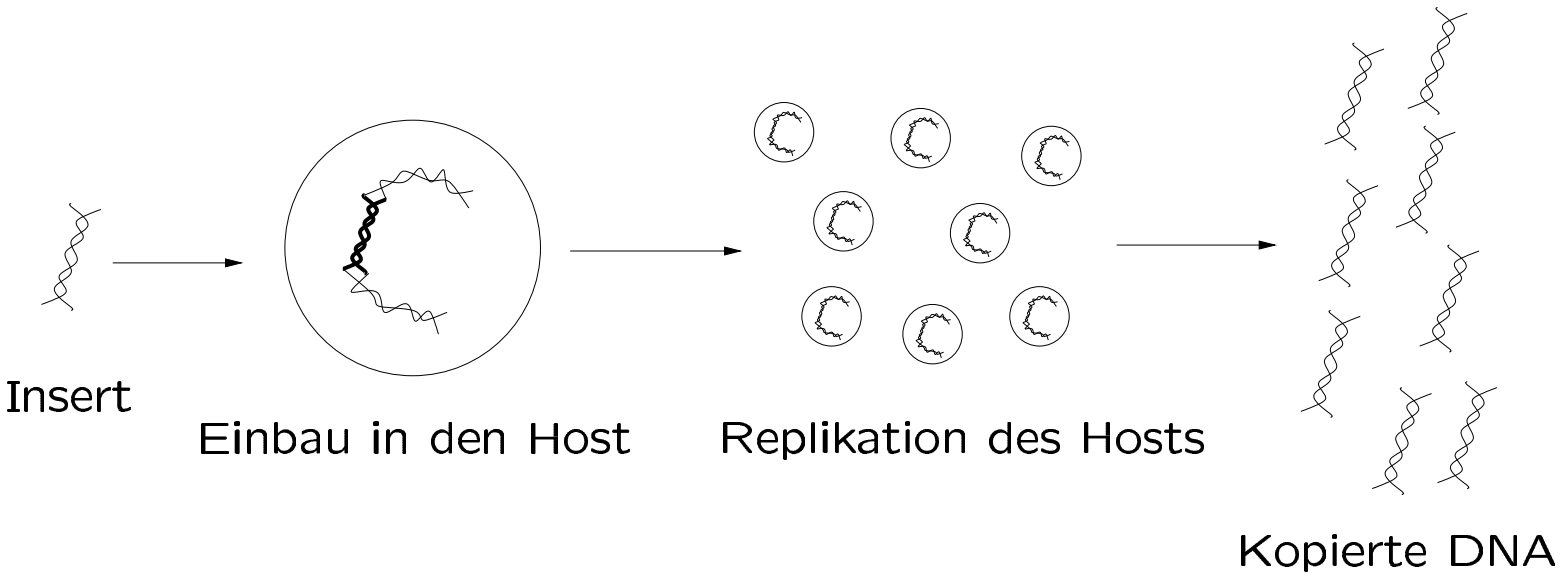


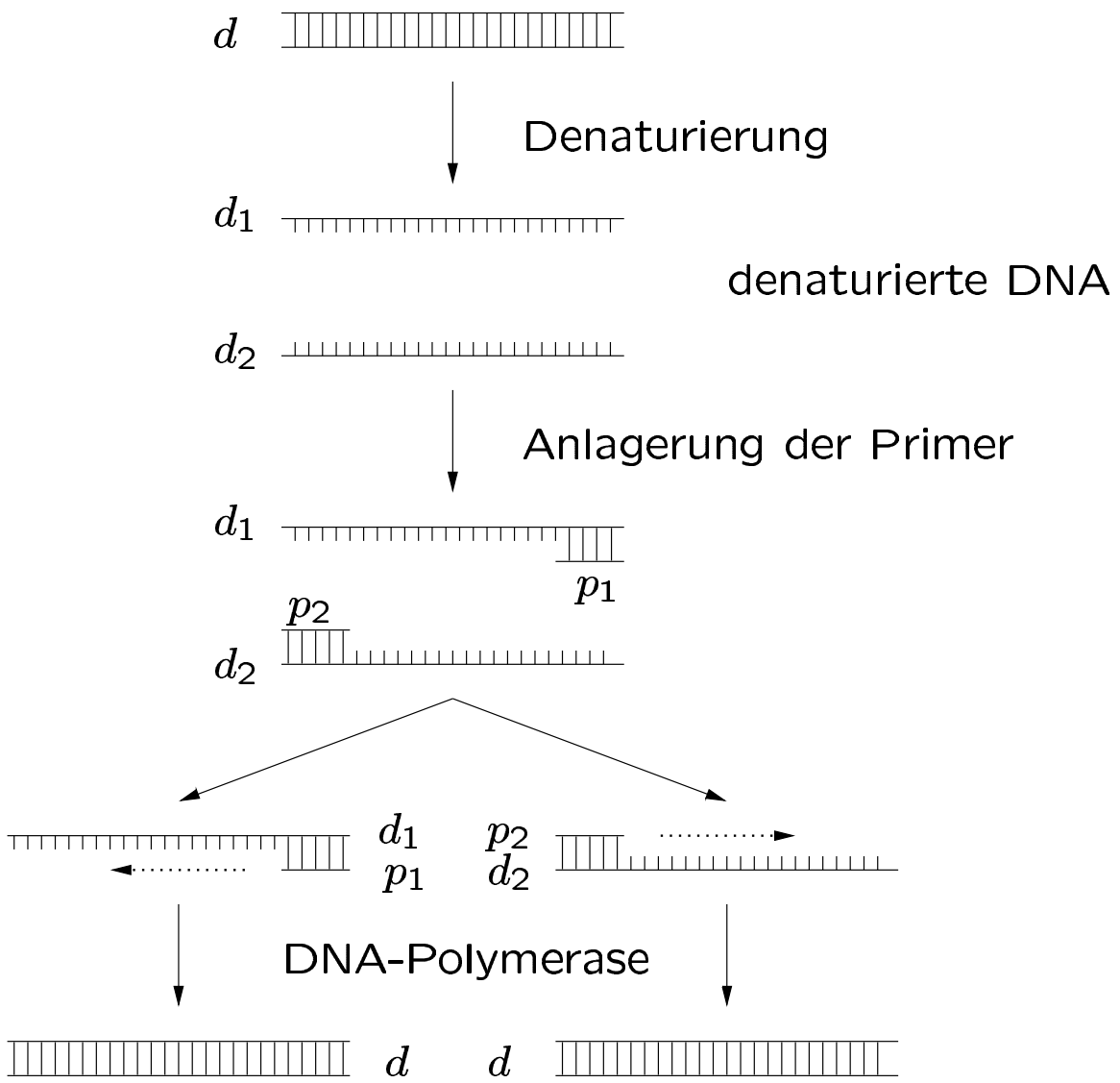
(h) $X = \{0, 7, 10, 14, 19\}$; $A = \{1, 2, 5, 9, 10\}$; $y = 10$; $S = (14, 7, 10)$;
$\delta(10, X) = \{0, 3, 4, 9, 10\} \not\subseteq A$;
$\delta(19 - 10, X) = \delta(9, X) = \{1, 2, 5, 9, 10\} \subseteq A$;
$\Rightarrow$ Platzierung: *rechts*



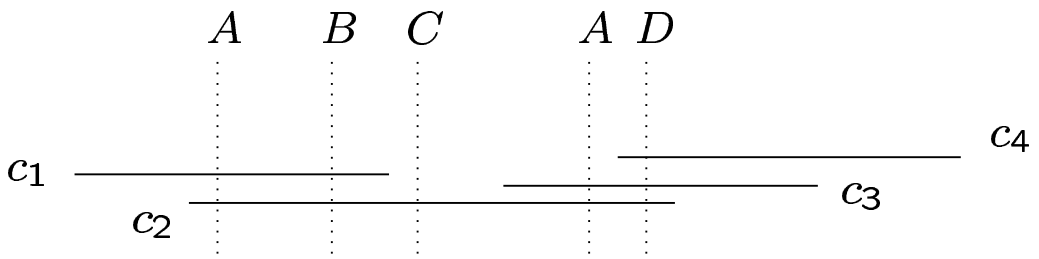
$P = \{0, 7, 9, 10, 14, 19\}$ ist eine zulässige Lösung und $\overline{P} = \{0, 5, 9, 10, 12, 19\}$.


{0, 19}
{0, 14, 19}
Backtrack
{0, 12, 14, 19}
{0, 7, 14, 19}
Backtrack
{0, 9, 12, 14, 19}
{0, 7, 10, 14, 19}
{0, 7, 9, 10, 14, 19}
zulässige Lösung

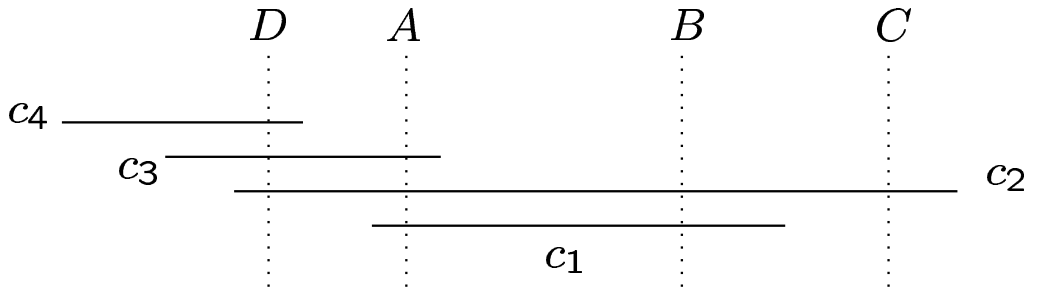

Insert
Einbau in den Host
Replikation des Hosts
Kopierte DNA


d
Denaturierung
d_1
denaturierte DNA
d_2
Anlagerung der Primer
d_1
p_1
p_2
d_2
d_1 p_2
p_1 d_2
DNA-Polymerase
d d

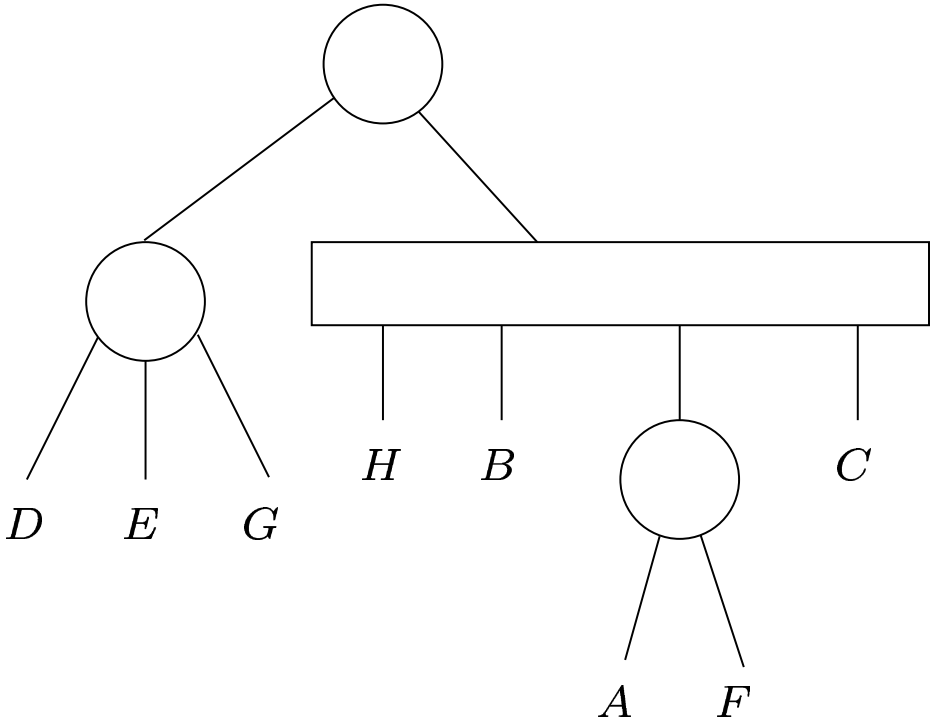
	A	B	C	D
c_1	1	1	0	0
c_2	1	1	1	1
c_3	1	0	0	1
c_4	0	0	0	1

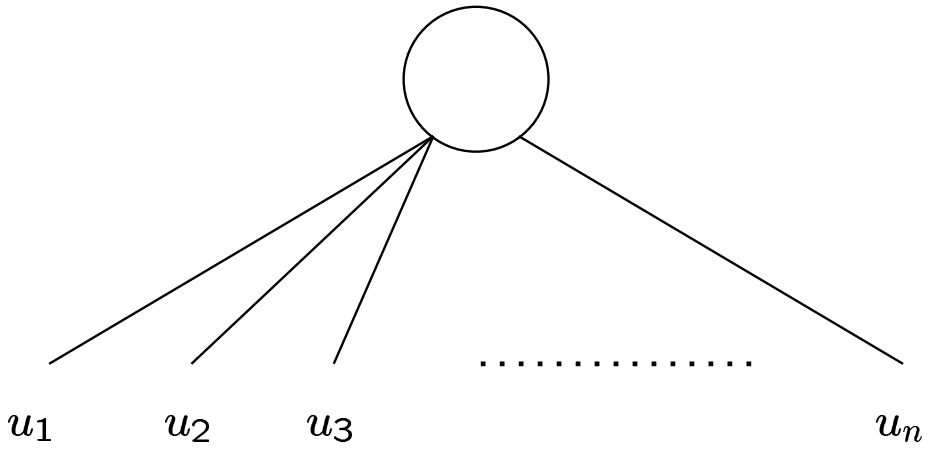


(a)

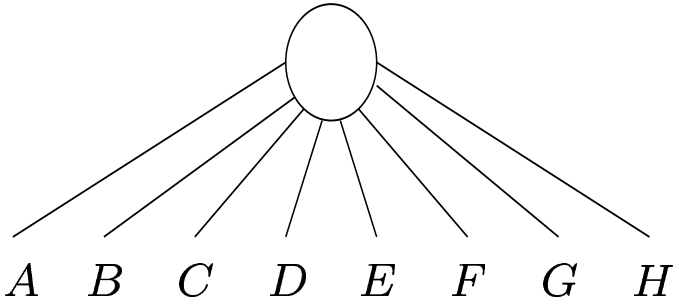


(b)


D
E
G
H
B
A
F
C


u_1
u_2
u_3
u_n

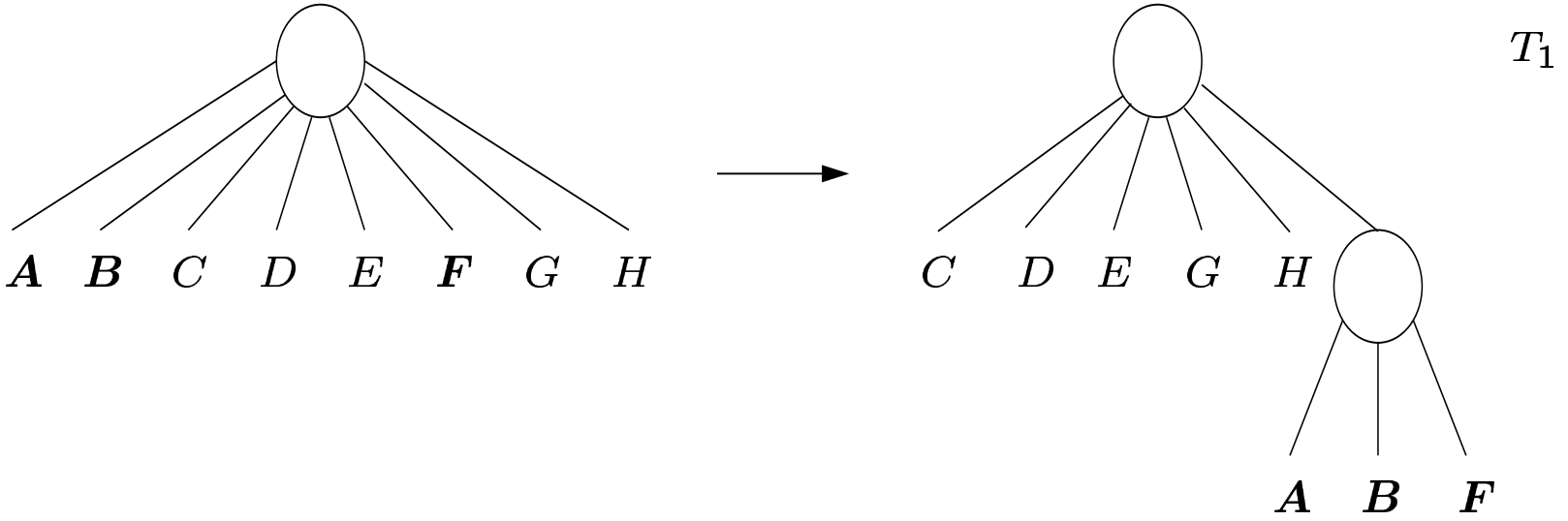
universeller PQ-Baum



T_0

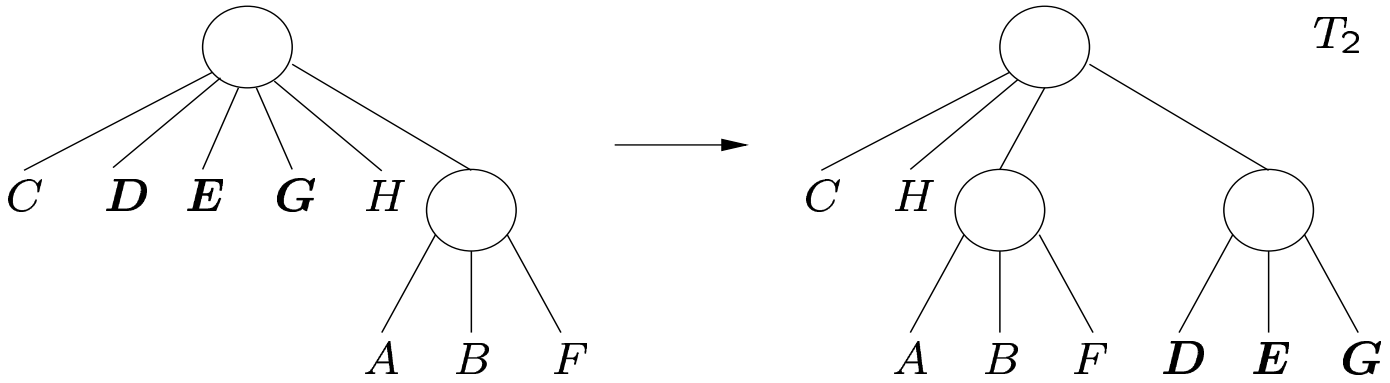
	A	B	C	D	E	F	G	H
I	1	1	0	0	0	1	0	0
II	0	0	0	1	1	0	1	0
III	0	1	0	0	0	0	0	1

Zeile I, $R_{\mathrm{I}} = \{A, B, F\}$:

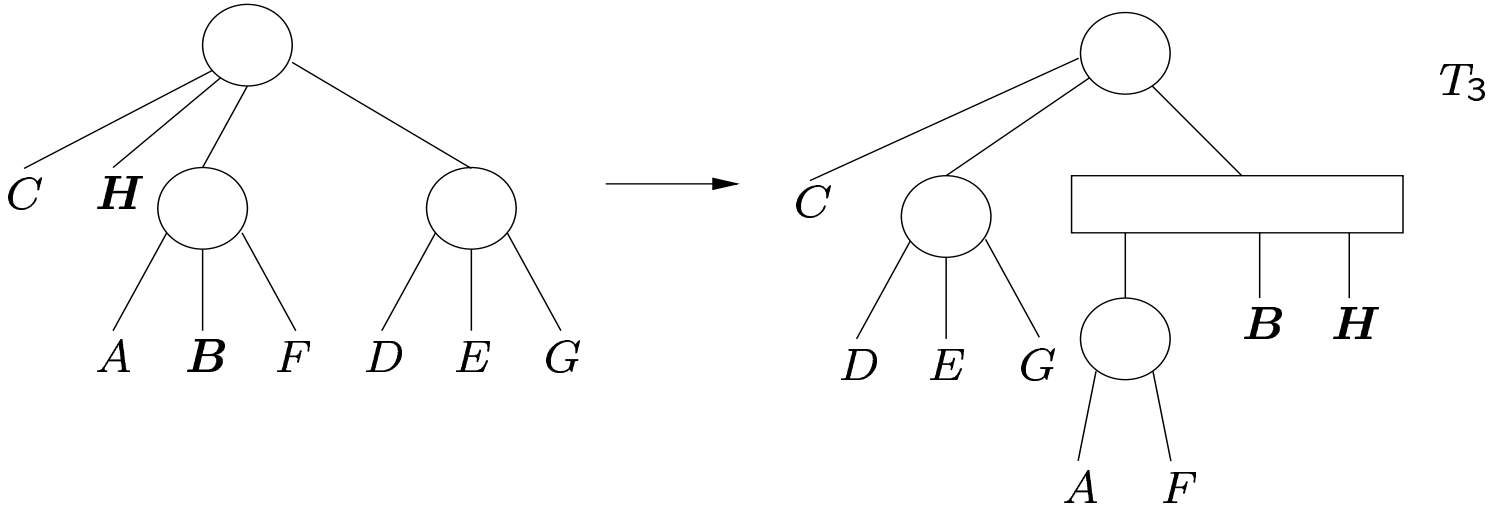


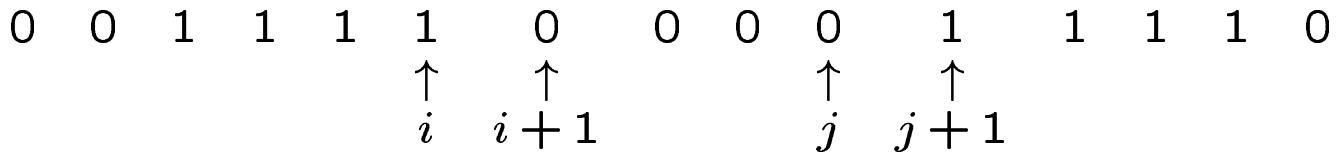
T_1

Zeile II, $R_{\text{II}} = \{D, E, G\}$:



Zeile III, $R_{\text{III}} = \{B, H\}$:




0 0 1 1 1 1 0 0 0 0 1 1 1 1 0
i
i + 1
j
j + 1

(a)	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1
(b)	0	0	1	1	1	1	0	0	0	0	1	1	1	0	0

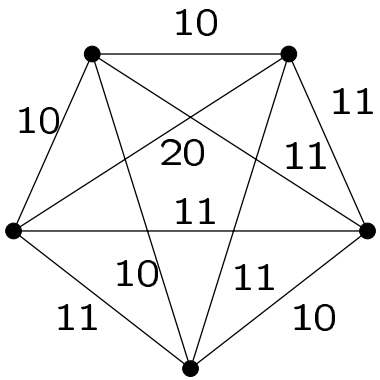
Eingabe: Ein ungerichteter vollständiger kantengewichteter Graph $G = (V, E, d)$ mit n Knoten und einer Kantenbewertung $d : E \to \mathbb{Q}^{\geq 0}$.

1. Bestimme einen minimalen Spannbaum T von G.

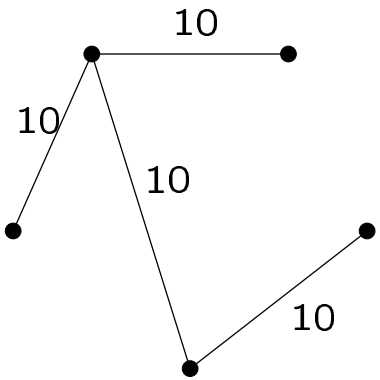
2. Führe in T eine Tiefensuche von einem beliebig gewählten Knoten $v \in V$ aus durch. Nummeriere hierbei die Knoten in der Reihenfolge, in der sie zum ersten Mal besucht werden. Sei $v_{i_1}, \ldots, v_{i_n}$ diese Reihenfolge.

3. Setze $H := v_{i_1}, \ldots, v_{i_n}, v_{i_1}$.

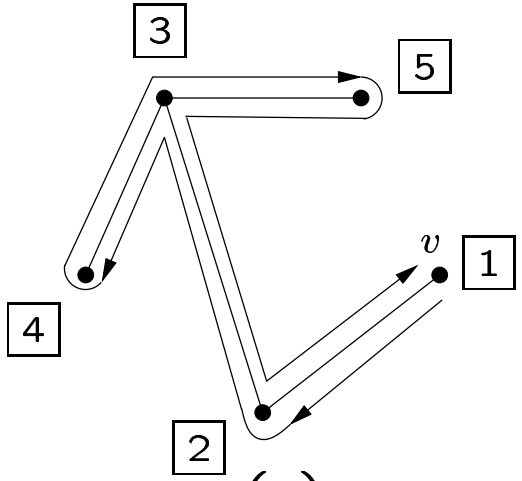
Ausgabe: Der Hamiltonkreis H von G.


10
10
11
20
11
11
10
11
11
10

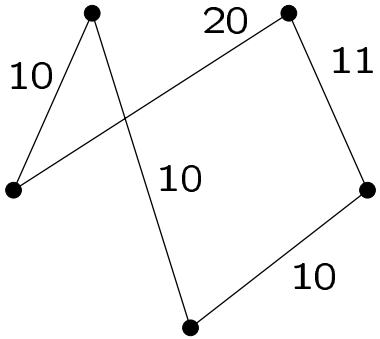
(a)


10
10
10
10

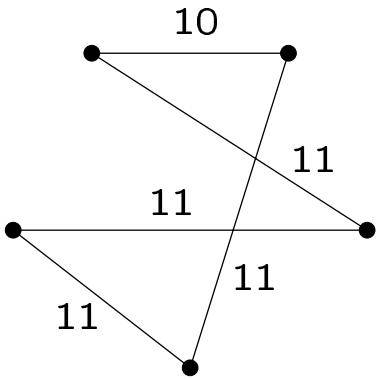
(b)


3
5
v
1
4
2

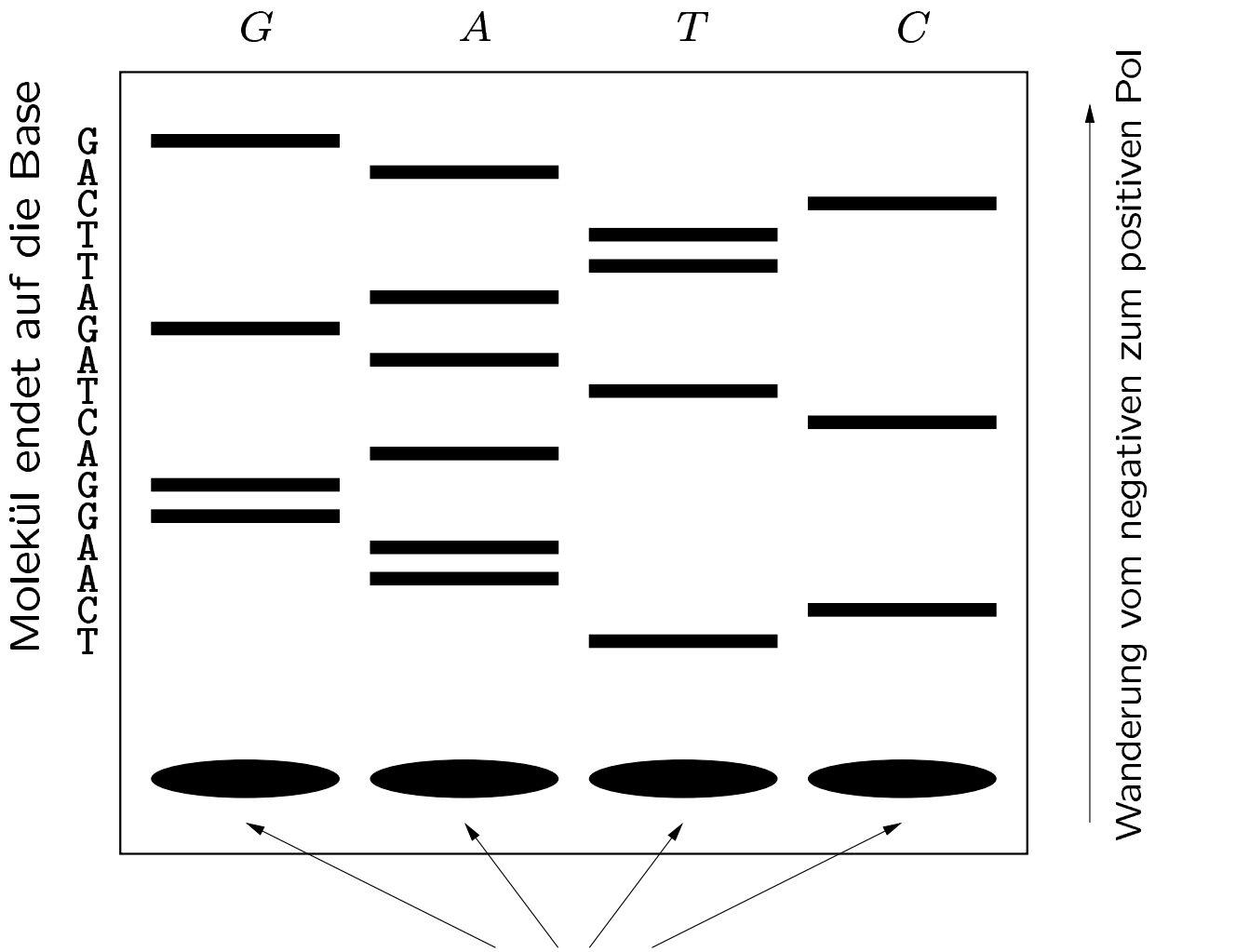
(c)

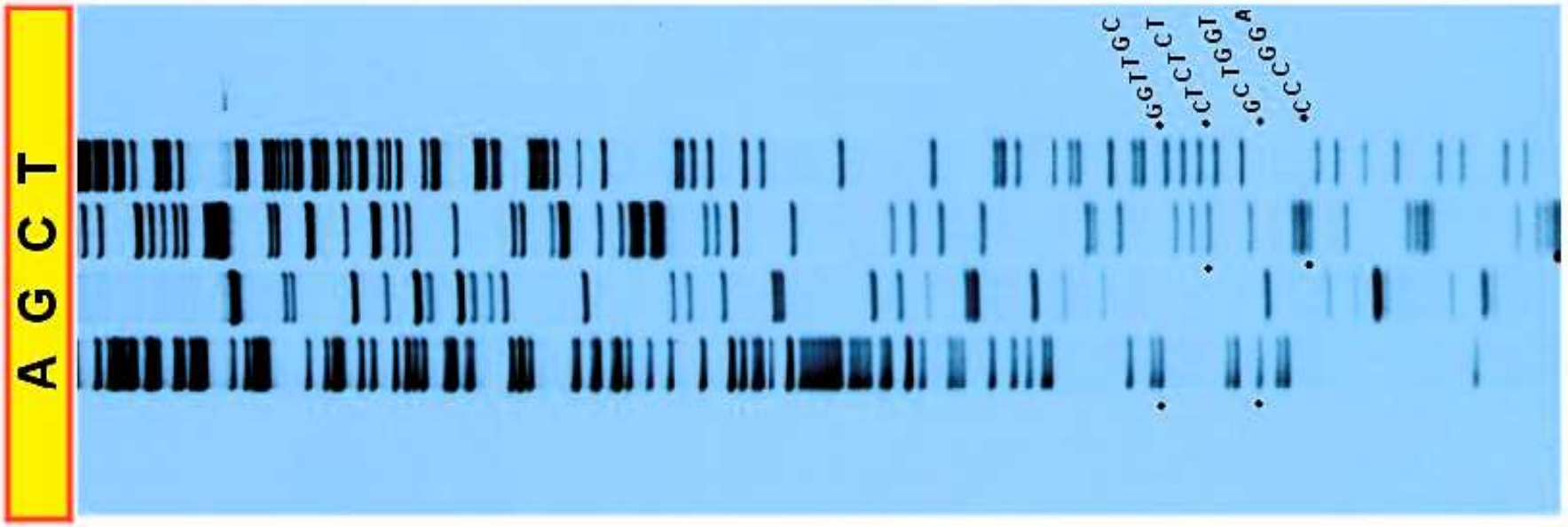

20
10
11
10
10

(d)

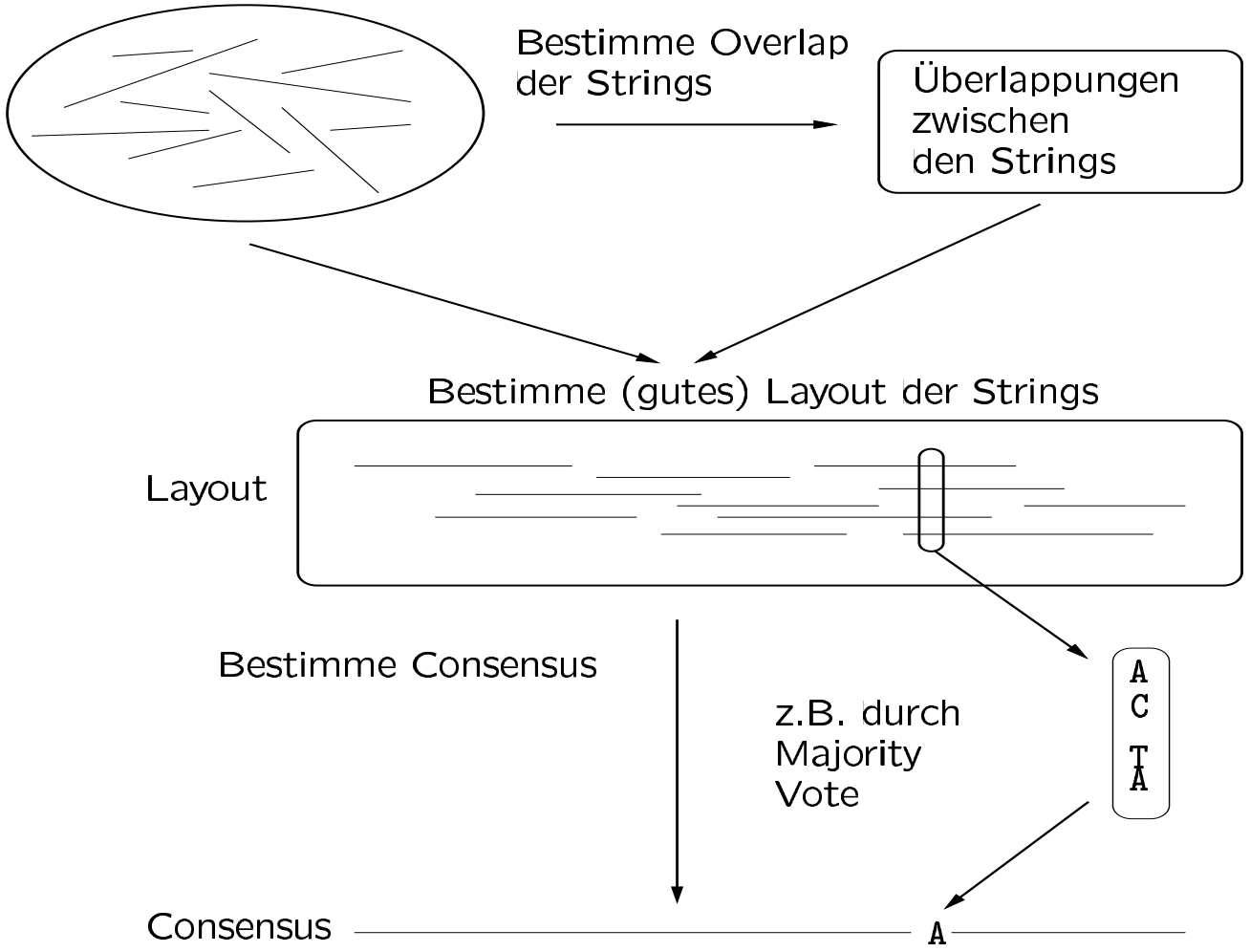

10
11
11
11
11

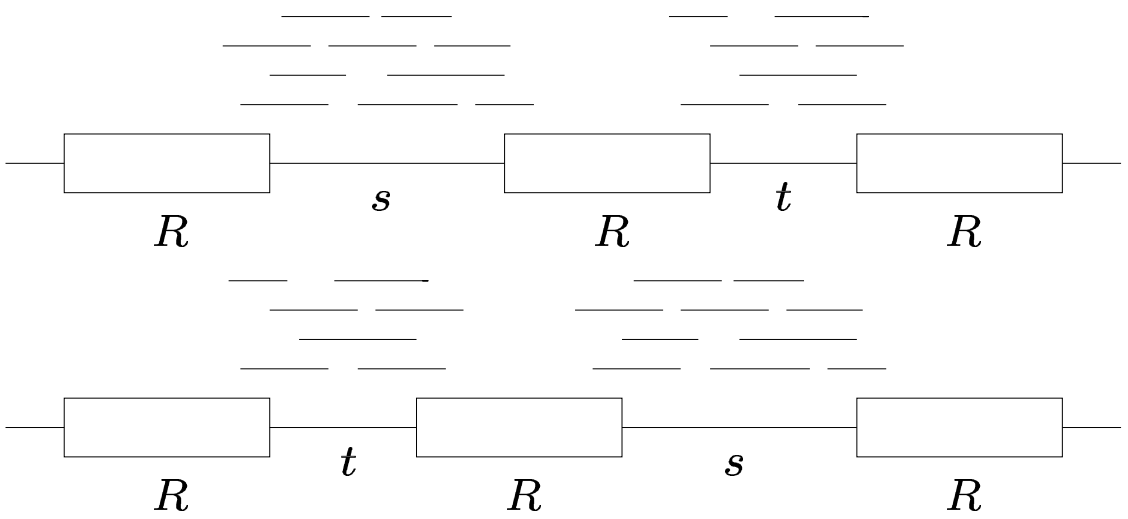
(e)


G
A
T
C
Molekül endet auf die Base
GACTTAGATCAGGAACT
Wanderung vom negativen zum positiven Pol
Auftragungspunkte der DNA-Fragmente auf das Gel


A G C T

Kollektion von DNA-Fragmenten,
gegeben als Strings




s
t
R
R
R
t
s
R
R
R

- $S = \{ababaa, bab, caba, aaddd, aabca, aacab\}$
- teilstringfrei :$S' = \{ababaa, caba, aaddd, aabca, aacab\}$.
- trivialer Superstring $w_T = ababaacabaaadddaabcaaacab$, Länge $|w_T| = 25$

$$\begin{array}{r} aacab \\ aabca \\ ababaa \\ caba \\ aaddd \\ \hline aadddcababaabcaacab \end{array}$$

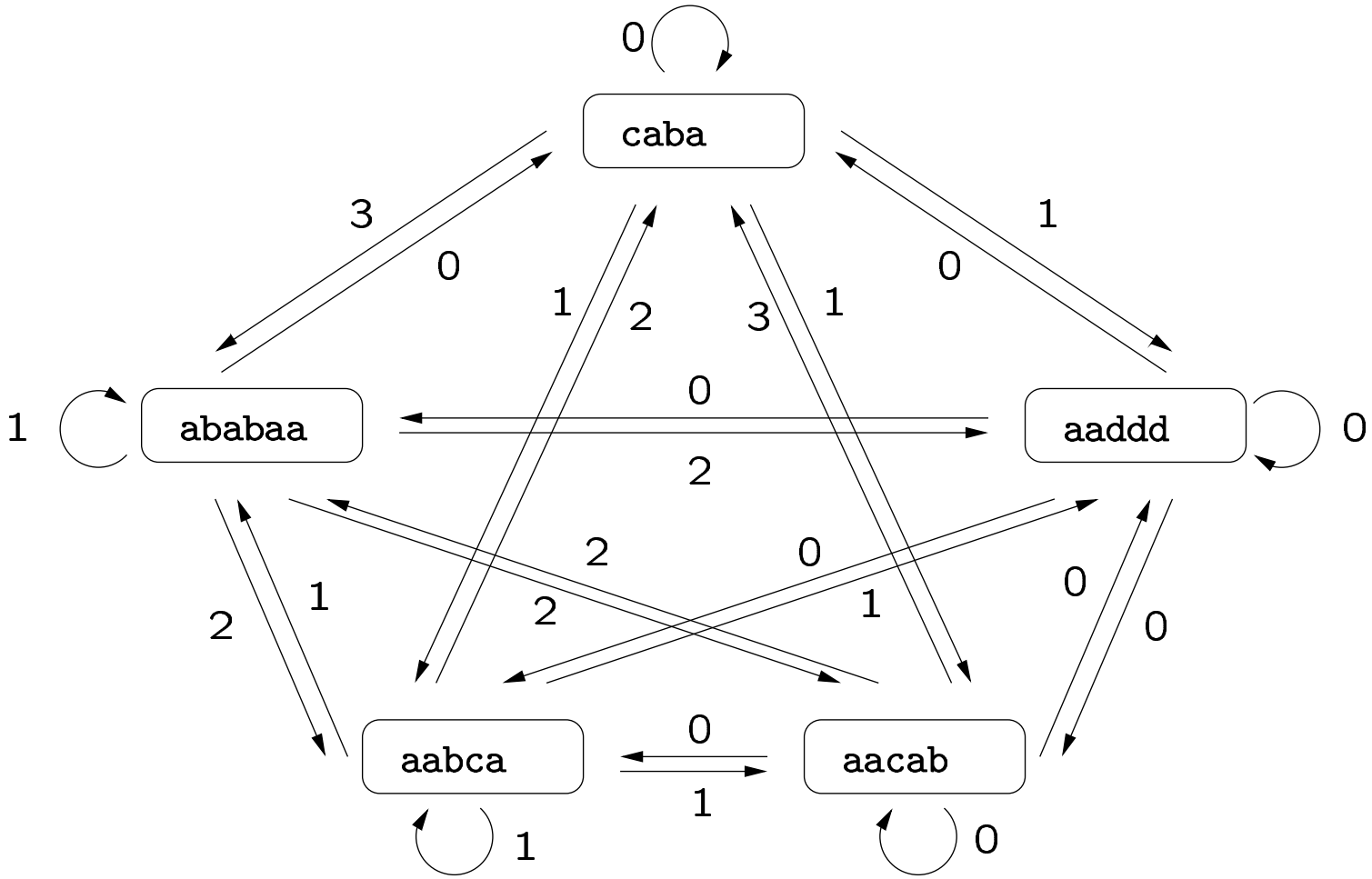
Länge 19, Kompression 6

$$\begin{array}{r} aaddd \\ aabca \\ ababaa \\ caba \\ aacab \\ \hline aacababaabcaaddd \end{array}$$

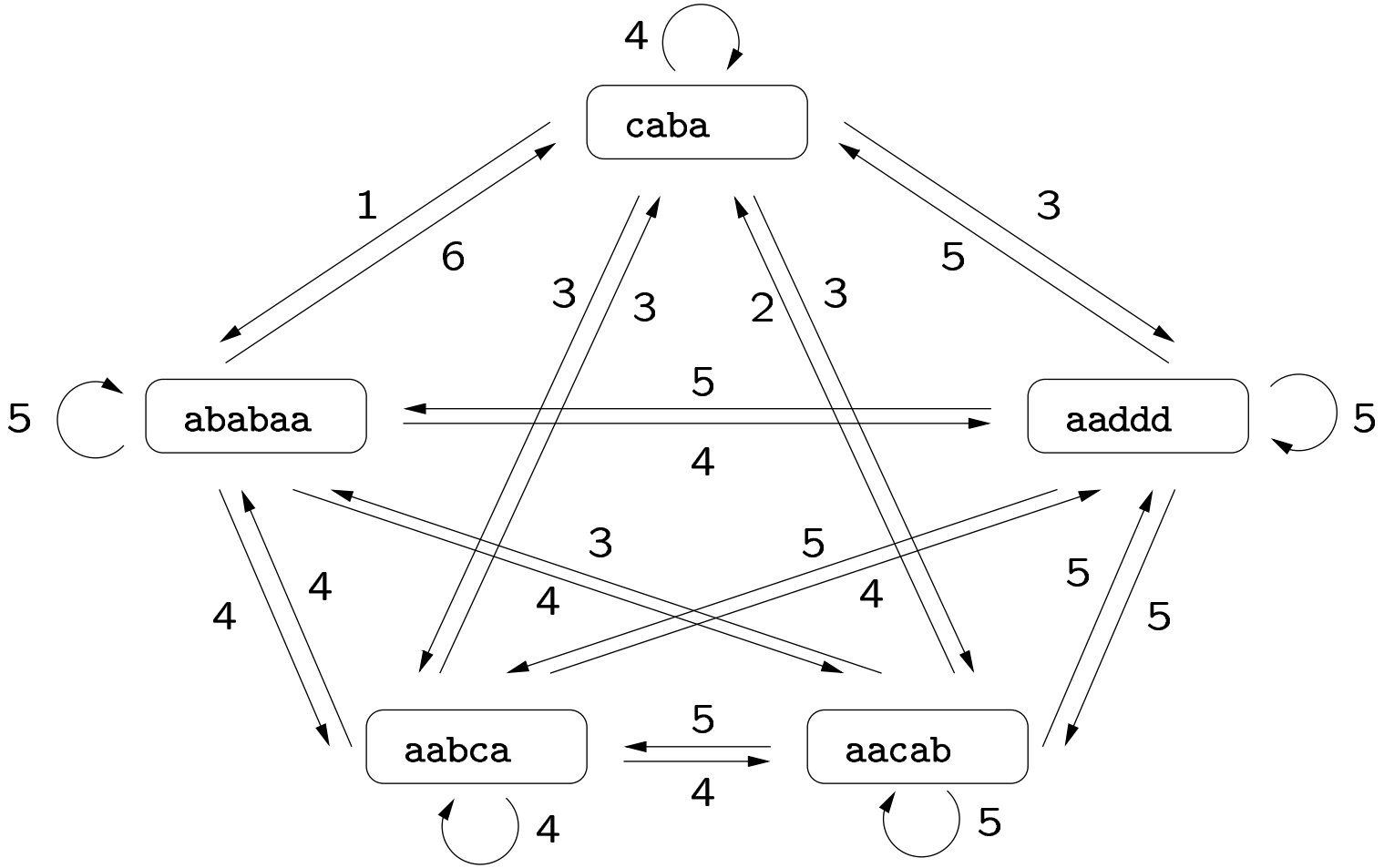
Länge 16, Kompression 9

Overlap-Graph

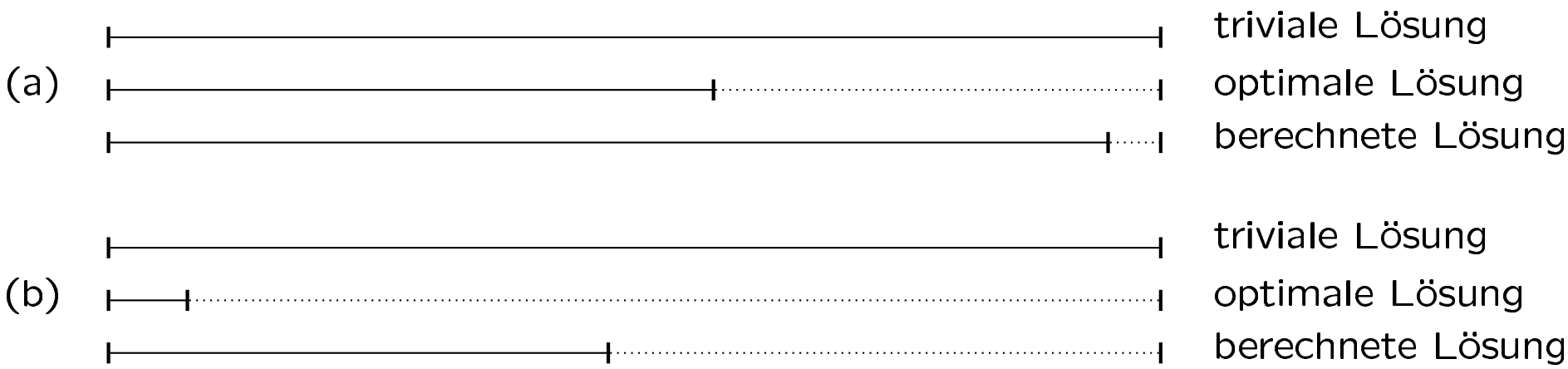
6

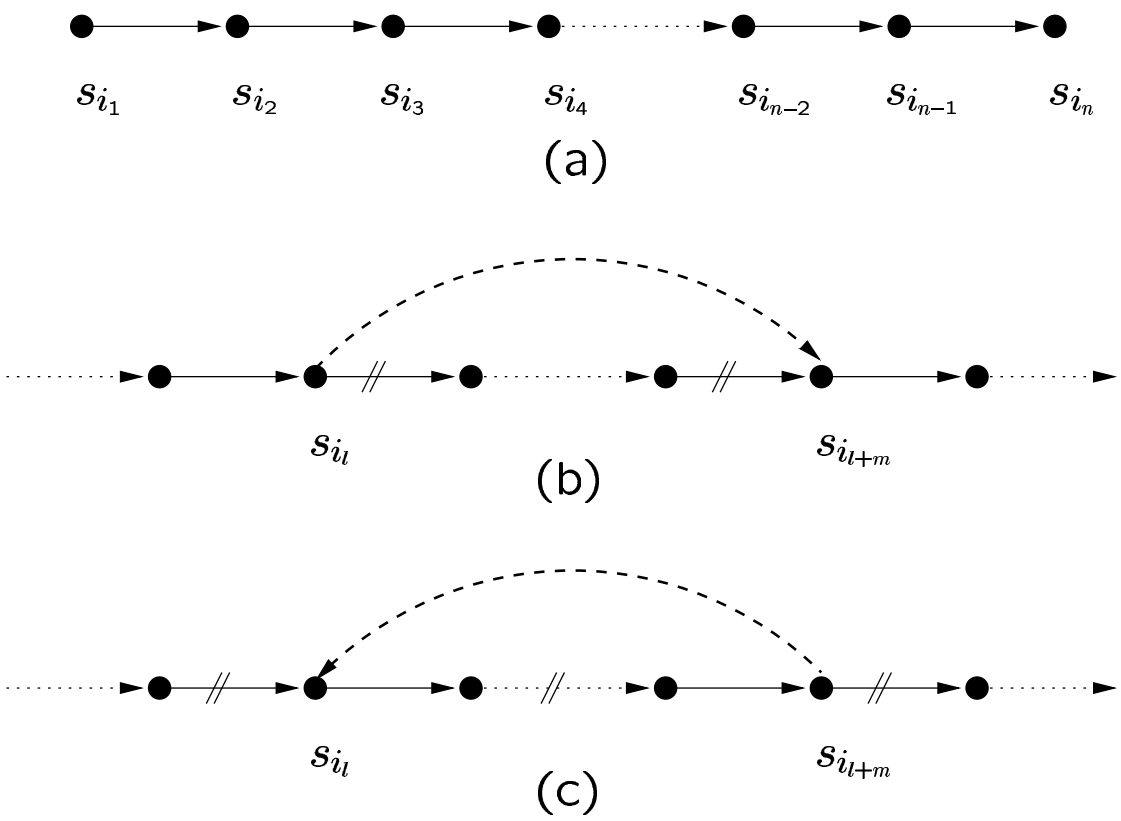


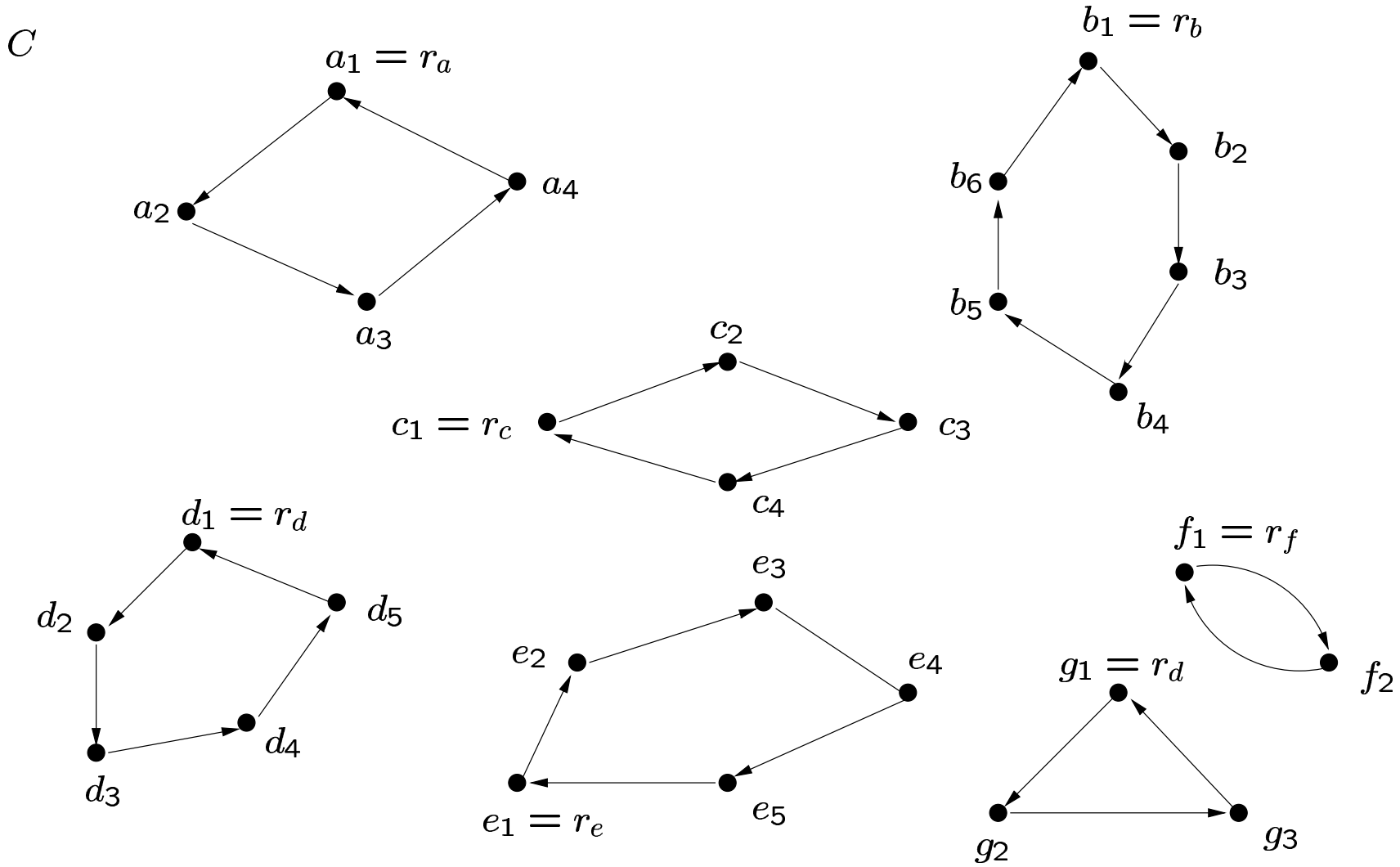
Distanz-Graph

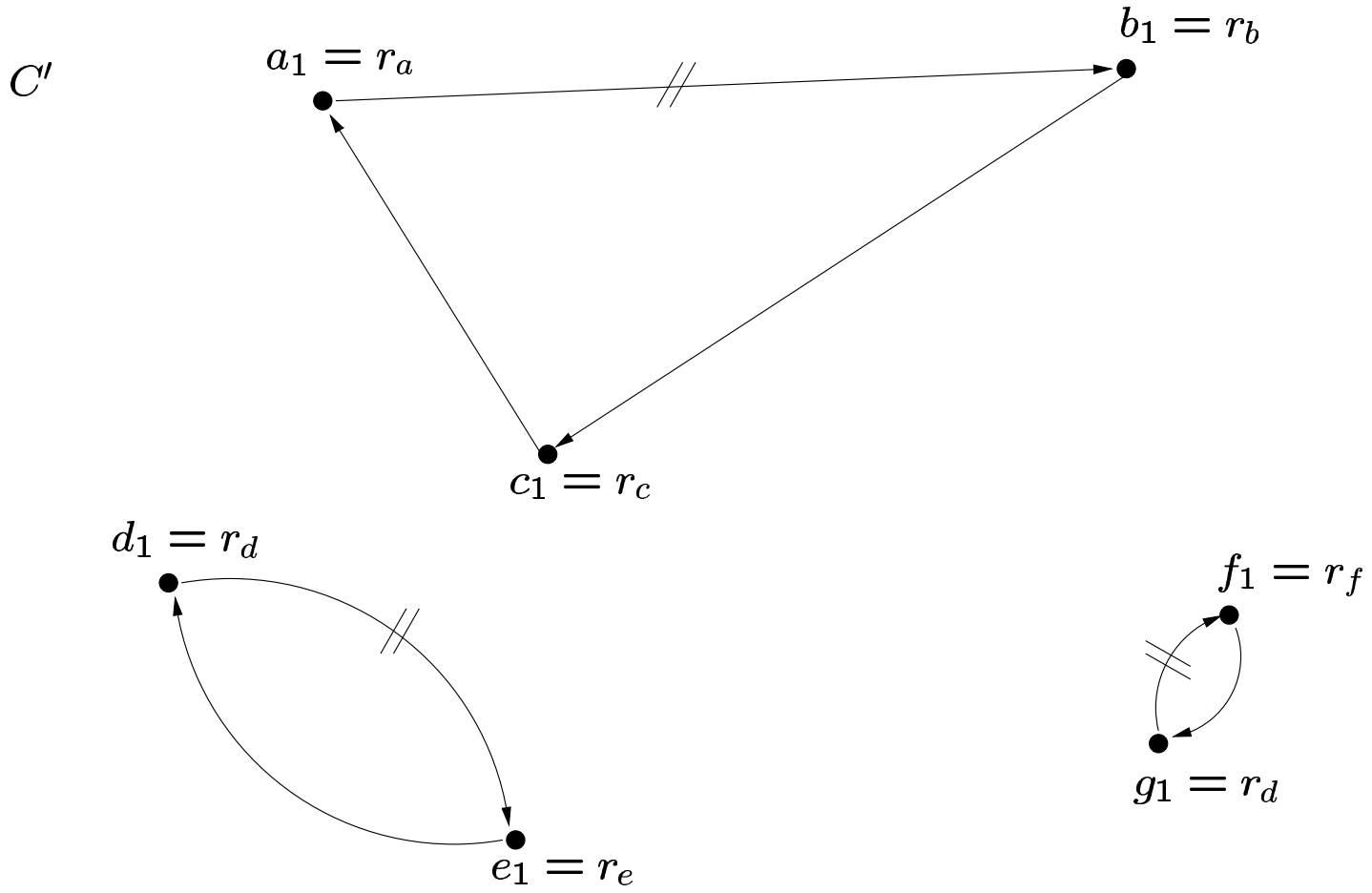


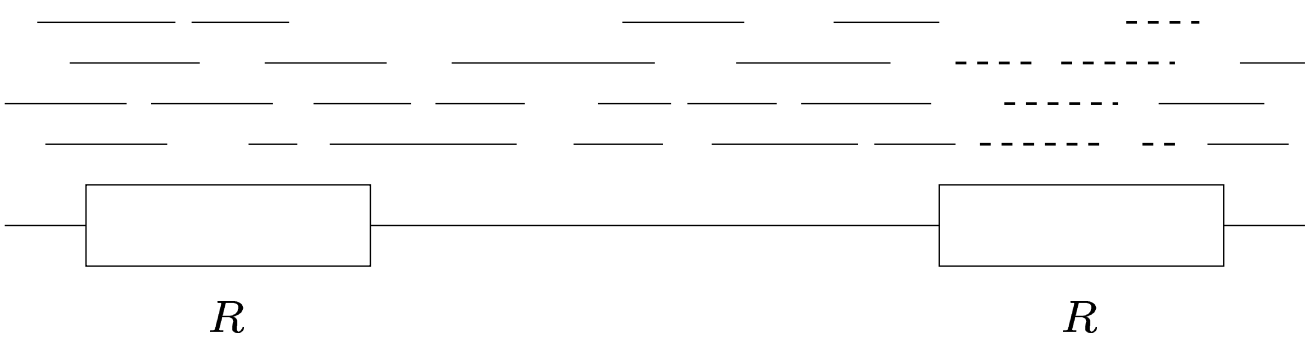
	$ababaa$	$caba$	$aaddd$	$aabca$	$aacab$
$ababaa$	1	0	**2**	2	2
$caba$	**3**	0	1	1	1
$aaddd$	0	0	0	0	0
$aabca$	1	2	1	1	**1**
$aacab$	2	**3**	0	0	0

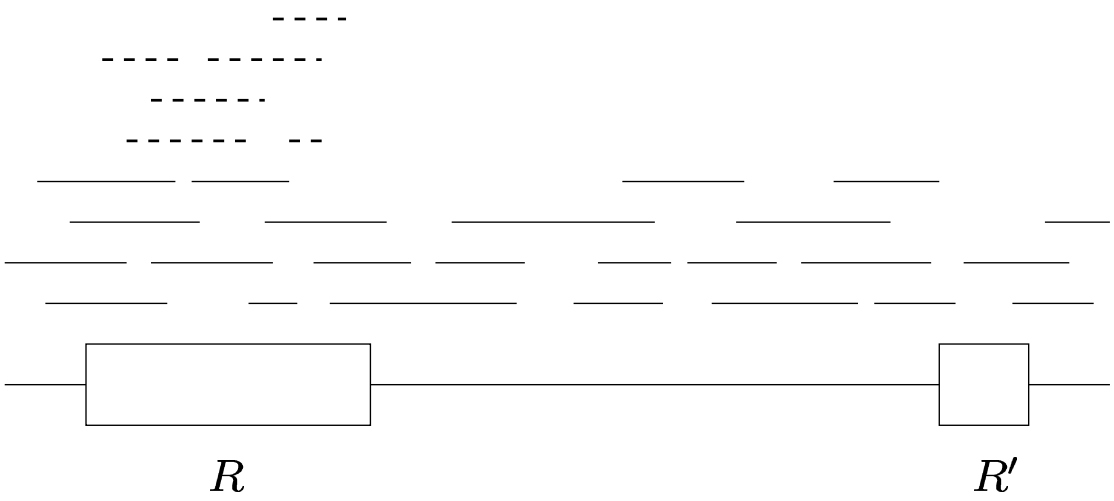

(a)
triviale Lösung
optimale Lösung
berechnete Lösung
(b)
triviale Lösung
optimale Lösung
berechnete Lösung


s_{i_1}
s_{i_2}
s_{i_3}
s_{i_4}
$s_{i_{n-2}}$
$s_{i_{n-1}}$
s_{i_n}
(a)
s_{i_l}
$s_{i_{l+m}}$
(b)
s_{i_l}
$s_{i_{l+m}}$
(c)

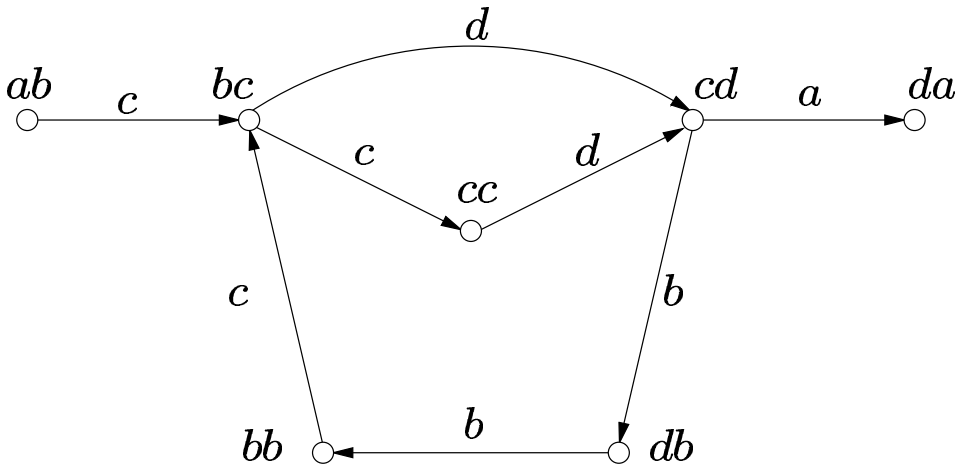

C
$a_1 = r_a$
a_2
a_3
a_4
$b_1 = r_b$
b_2
b_3
b_4
b_5
b_6
$c_1 = r_c$
c_2
c_3
c_4
$d_1 = r_d$
d_2
d_3
d_4
d_5
$e_1 = r_e$
e_2
e_3
e_4
e_5
$f_1 = r_f$
f_2
$g_1 = r_d$
g_2
g_3

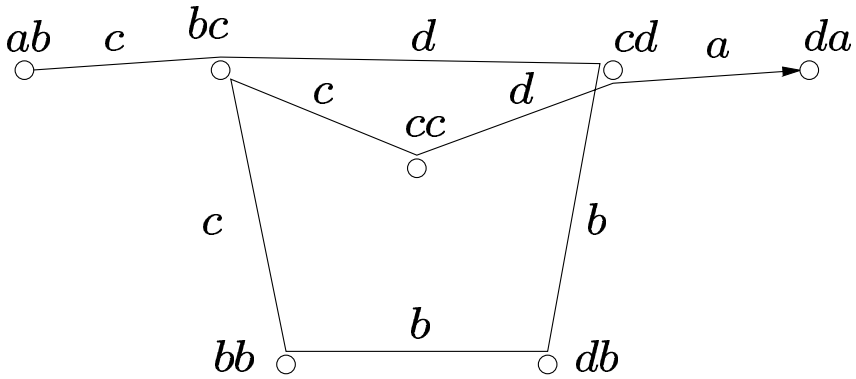

C'
$a_1 = r_a$
$b_1 = r_b$
$c_1 = r_c$
$d_1 = r_d$
$e_1 = r_e$
$f_1 = r_f$
$g_1 = r_d$


R
R

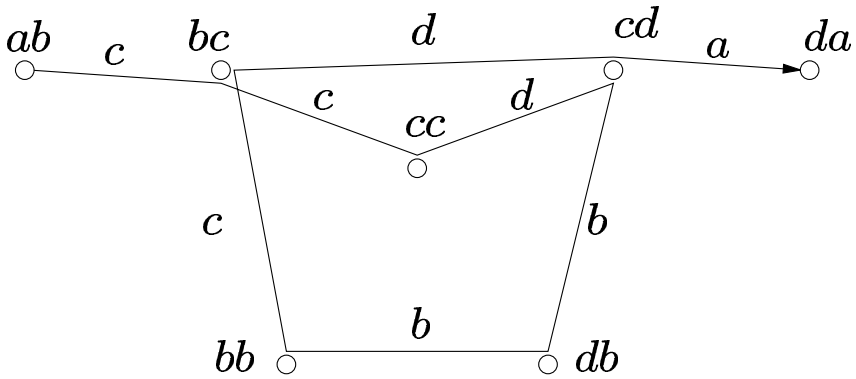

R
R′

Spektrum $S = \{abc, bbc, bcc, bcd, ccd, cda, cdb, dbb\}$

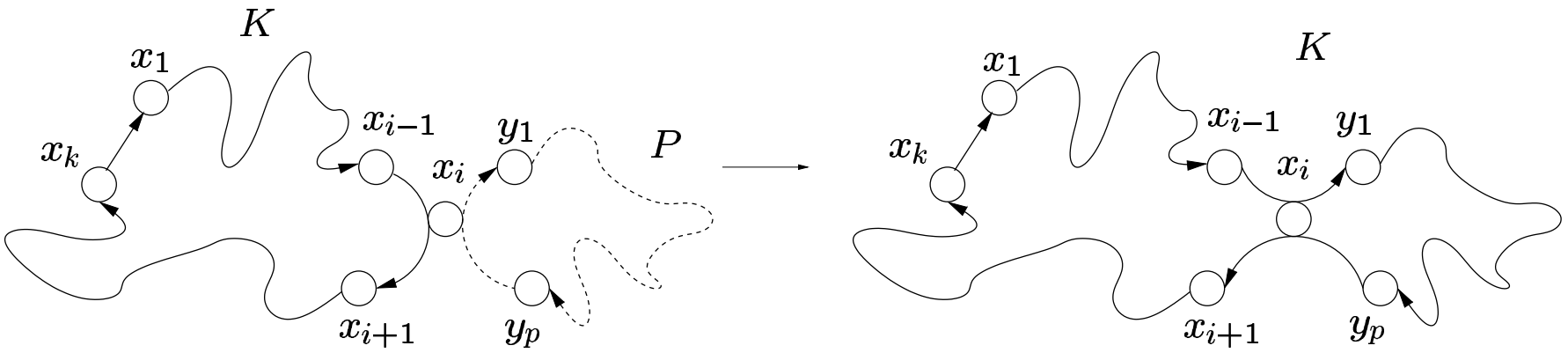




$w_1 = abcdbbccda$



$w_2 = abccdbbcda$


K
x_1
x_k
x_{i-1}
x_i
y_1
P
x_{i+1}
y_p
K
x_1
x_k
x_{i-1}
y_1
x_i
x_{i+1}
y_p

i	$t_1 \dots t_{l-i}$	$t_{i+1} \dots t_l$	$c_i^{(t)}$
0	$ababa$	$ababa$	1
1	$abab$	$baba$	0
2	aba	aba	1
3	ab	ba	0
4	a	a	1

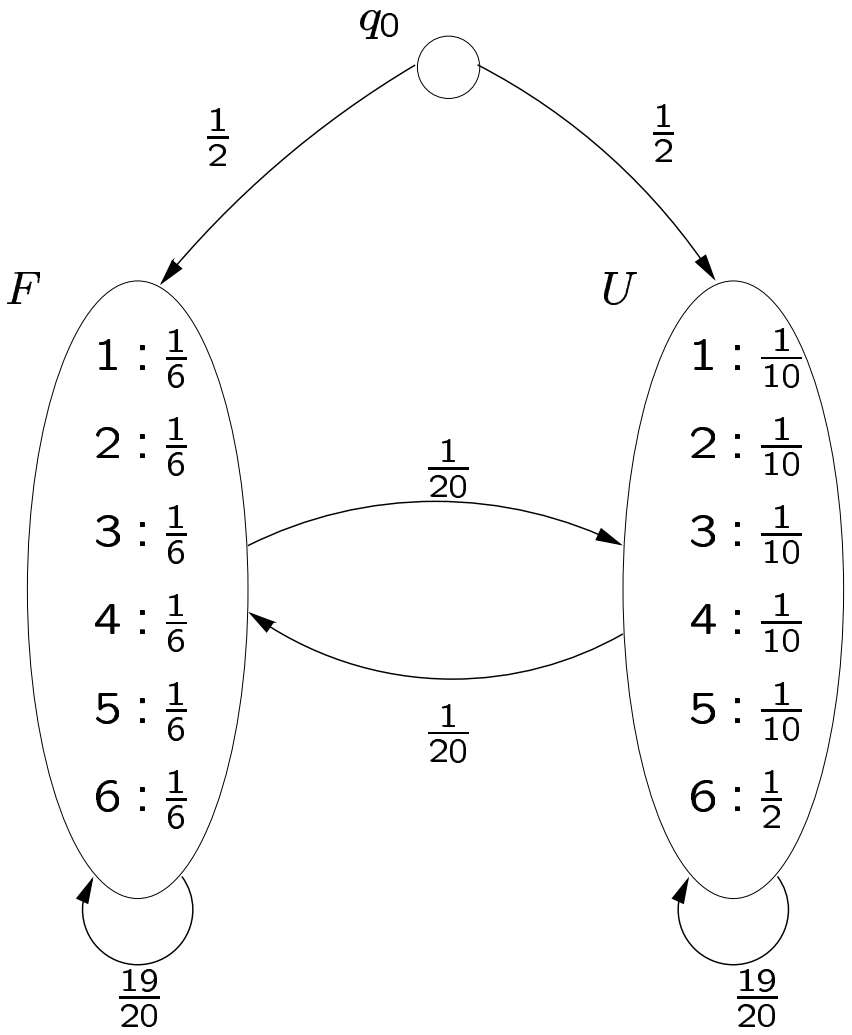
$$corr_t(x) = 1 + x^2 + x^4.$$

- $\Sigma = \{1, 2, 3, 4, 5, 6\}$
- $Q = \{q_0, F, U\}$
-

δ	q_0	F	U
q_0	0	$\frac{1}{2}$	$\frac{1}{2}$
F	0	$\frac{19}{20}$	$\frac{1}{20}$
U	0	$\frac{1}{20}$	$\frac{19}{20}$

-

η	1	2	3	4	5	6
F	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
U	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{2}$



Eingabe: Ein HMM $\mathcal{M} = (\Sigma, Q, q_0, \delta, \eta)$ und ein String $x = x_1, \ldots, x_n \in \Sigma^n$.

1. Initialisierung:

   $\sigma_{q_0}(0) := 1$
   for all $q \in Q - \{q_0\}$ **do**
   $\sigma_q(0) := 0$

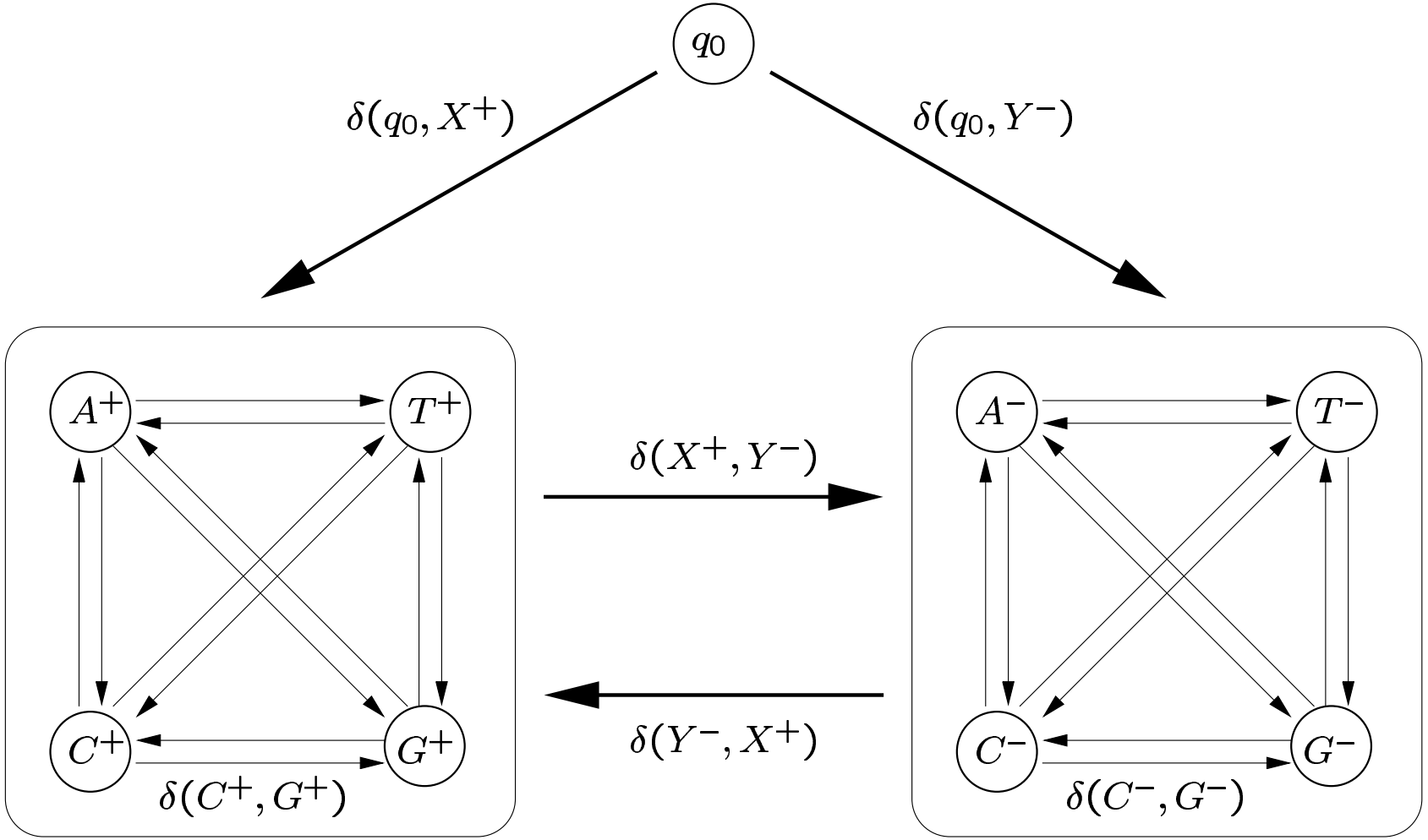
2. Berechnung der $\sigma_q(i)$:

   for $i = 1$ **to** n **do**
   for all $q \in Q - \{q_0\}$ **do**
   $\sigma_q(i) := \eta(q, x_i) \cdot \max_{p \in Q}(\sigma_p(i-1) \cdot \delta(p, q))$
   $ptr_q(i) := \text{argmax}_{p \in Q}(\sigma_p(i-1) \cdot \delta(p, q))$

3. Traceback zur Bestimmung eines wahrscheinlichsten Pfades π^*:

   $Prob[x \mid \pi^*] := \max_{p \in Q - \{q_0\}}(\sigma_p(n))$
   $\pi_n^* := \text{argmax}_{p \in Q - \{q_0\}}(\sigma_p(n))$
   for $i := n - 1$ **downto** 0 **do**
   $\pi_i^* := ptr_{\pi_{i+1}^*}(i+1)$

Ausgabe: $\pi^* = \pi_0^*, \ldots, \pi_n^*$ als einen wahrscheinlichsten Pfad für x in M mit der Wahrscheinlichkeit $Prob[x \mid \pi^*]$.

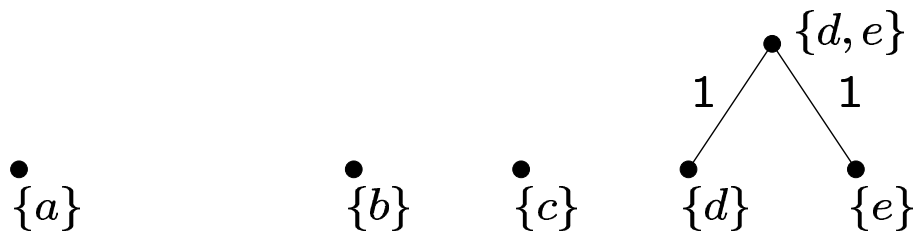

q_0
$\delta(q_0, X^+)$
$\delta(q_0, Y^-)$
A^+
T^+
C^+
G^+
$\delta(C^+, G^+)$
$\delta(X^+, Y^-)$
$\delta(Y^-, X^+)$
A^-
T^-
C^-
G^-
$\delta(C^-, G^-)$

Eingabe: Menge $A = \{a_1, \ldots, a_n\}$ von Taxa und ultrametrische Distanzfunktion $\delta : A \times A \to \mathbb{Q}^{\geq 0}$.

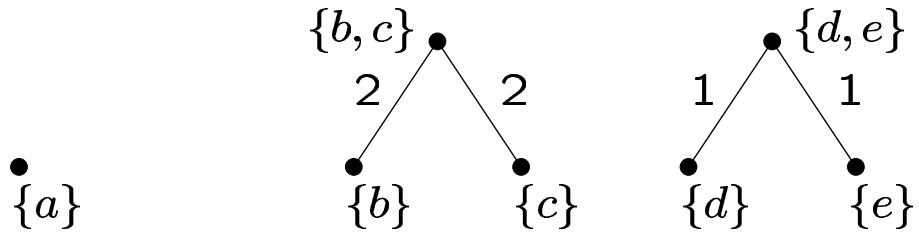
1. (a) $\Gamma := \{\{a_1\}, \ldots, \{a_n\}\}$, $V := \Gamma$ und $E := \emptyset$.
   (b) $dist(\{a_i\}, \{a_j\}) := \delta(a_i, a_j)$ für alle $i, j \in \{1, \ldots, n\}$.
   (c) $height(\{a_i\}) := 0$ für alle $i \in \{1, \ldots, n\}$.
2. **while** $|\Gamma| \geq 2$ **do**
   (a) Finde $C_1, C_2 \in \Gamma$, $C_1 \neq C_2$, so dass $dist(C_1, C_2)$ minimal ist, und setze $D := C_1 \cup C_2$.
   (b) $\Gamma := (\Gamma - \{C_1, C_2\}) \cup \{D\}$.
   (c) $dist(D, X) = dist(X, D) := \dfrac{dist(C_1, X) + dist(C_2, X)}{2}$ für alle $X \in \Gamma$.
   (d) $V := V \cup \{D\}$ und $E := E \cup \{(D, C_1), (D, C_2)\}$.
   (e) $height(D) := \dfrac{dist(C_1, C_2)}{2}$.
   (f) $d(D, C_i) := height(D) - height(C_i)$ für $i = 1, 2$

Ausgabe: Der ultrametrische Baum $T = (V, E, d)$ für A.

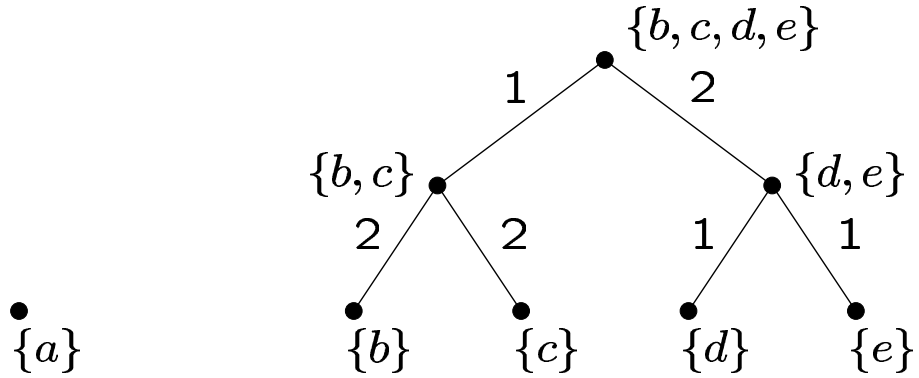
$dist$	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$
$\{a\}$	0	12	12	12	12
$\{b\}$	12	0	4	6	6
$\{c\}$	12	4	0	6	6
$\{d\}$	12	6	6	0	2
$\{e\}$	12	6	6	2	0



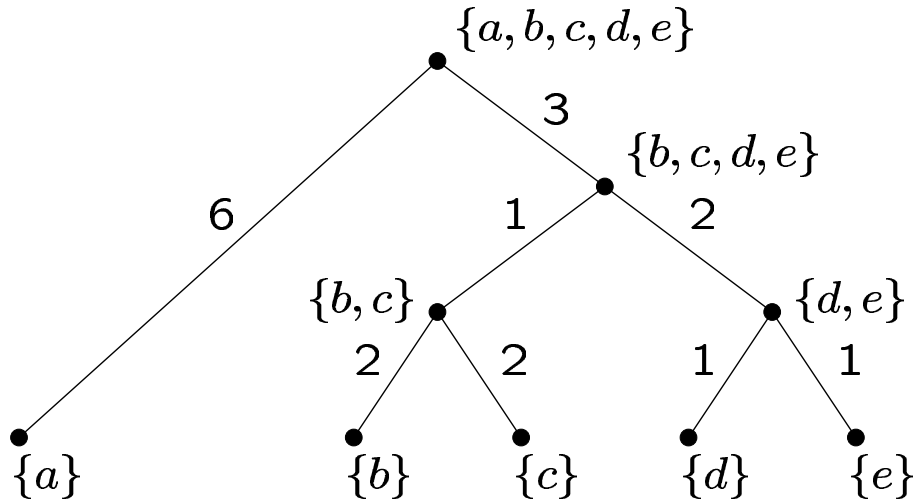
$dist$	$\{a\}$	$\{b\}$	$\{c\}$	$\{d,e\}$
$\{a\}$	0	12	12	12
$\{b\}$	12	0	4	6
$\{c\}$	12	4	0	6
$\{d,e\}$	12	6	6	0

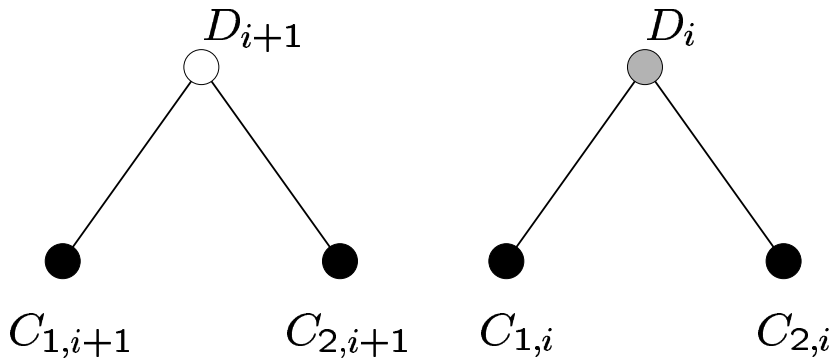


$dist$	$\{a\}$	$\{b,c\}$	$\{d,e\}$
$\{a\}$	0	12	12
$\{b,c\}$	12	0	6
$\{d,e\}$	12	6	0

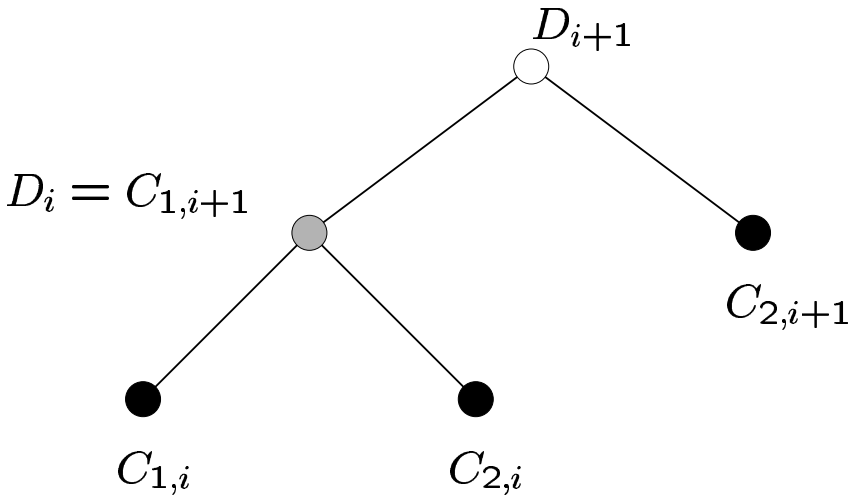


$dist$	$\{a\}$	$\{b,c,d,e\}$
$\{a\}$	0	12
$\{b,c,d,e\}$	12	0





(a)



(b)

Eingabe: Menge $A = \{a_1, \ldots, a_n\}$ von Taxa und metrisches Distanzmaß δ auf A.

1. Bestimme den Distanz-Graphen $G(A, \delta) = (V, E')$.

2. Initialisiere den Baum $T = (U, E, \delta)$ mit $U := \{a_1\}$ und $E := \emptyset$.

3. **while** $U \neq V$ **do**

   (a) Bestimme die Kante $\{x, y\}$ in $G(A, \delta)$ mit dem kleinsten Gewicht, so dass $x \in U$ und $y \in V - U$ gilt.

   (b) **if** gewählte Kante nicht eindeutig **then**
   Ausgabe „Es existiert kein kompakter additiver Baum“; **stop**

   (c) **if** Additivitätsbedingung für y und alle Knoten in U nicht erfüllt **then**
   Ausgabe „Es existiert kein kompakter additiver Baum“; **stop**

   (d) $U := U \cup \{y\}$ und $E := E \cup \{\{x, y\}\}$.

Ausgabe: Der kompakte additive Baum $T = (U, E, \delta)$.

Eingabe: Menge S von n Strings der Länge k über einem Alphabet Σ und ein ungerichteter phylogenetischer Baum $T = (V, E)$ für S.

1. Konstruiere Baum $T' = (V', E')$ mit Wurzel $r \notin V$ durch $V' := V \cup \{r\}$ und $E' := (E - \{\{x, y\}\}) \cup \{\{x, r\}, \{y, r\}\}$ für beliebige Kante $\{x, y\} \in E$.

2. $R(x, l) := \emptyset$ für alle $x \in V'$ und alle Positionen $l \in \{1, \ldots, k\}$; $cost := 0$

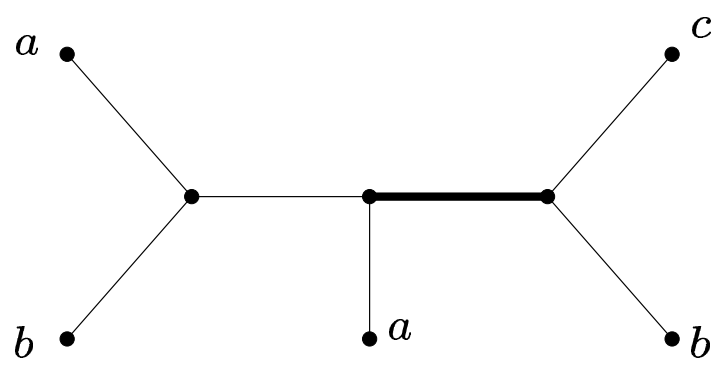
3. **for** $l := 1$ **to** k **do**
   Rufe die Prozedur $Fitch(r, l)$ zur Berechnung von $R(x, l)$ auf.

4. - Wähle für die Wurzel r eines ihrer Kinder y aus und setze für alle Positionen $l \in \{1, \ldots, k\}$
     $$\beta(r, l) := a \text{ für ein beliebiges } a \in R(y, l).$$
   - Durchlaufe T' von der Wurzel zu den Blättern und setze für jeden Knoten y mit dem Vater x und für jede Position $l \in \{1, \ldots, k\}$
     $$\beta(y, l) := \begin{cases} \beta(x, l) & \text{falls } \beta(x, l) \in R(y, l), \\ a \in R(y, l) \text{ beliebig} & \text{sonst.} \end{cases} \quad (1)$$
   - $\beta(x) := \beta(x, 1) \ldots \beta(x, k)$ für alle $x \in V$ und $cost(\beta) := cost$.

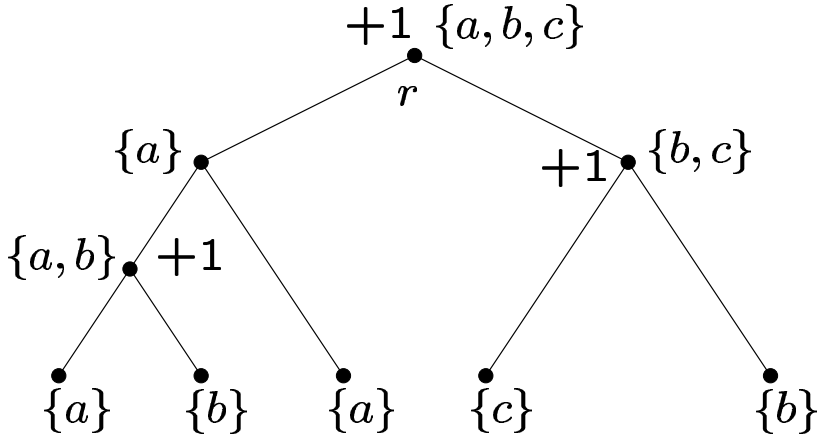
Ausgabe: Die Strings $\beta(x)$ für alle Knoten $x \in V$ und die Kosten $cost(\beta)$ dieser Beschriftung.

Prozedur $Fitch(x, l)$**:**

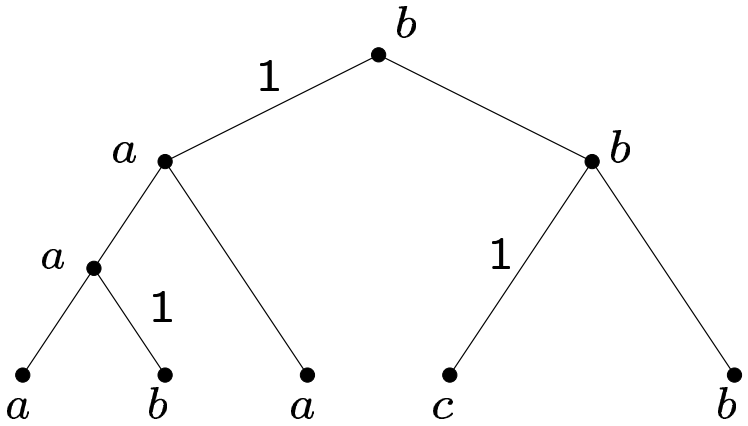
1. **if** x ist Blatt **then**
   Bestimme die Beschriftung $t = t_1 \dots t_k$ von x und setze $R(x, l) := t_l$.
2. **if** x ist innerer Knoten **then**
   - Bestimme die beiden Kinder y und z von x
   - Rufe die Prozeduren $Fitch(y, l)$ und $Fitch(z, l)$ auf.
   - **if** $R(y, l) \cap R(z, l) \neq \emptyset$ **then**
     $R(x, l) := R(y, l) \cap R(z, l)$
   - **if** $R(y, l) \cap R(z, l) = \emptyset$ **then**
     $cost := cost + 1$
     $R(x, l) := R(y, l) \cup R(z, l)$



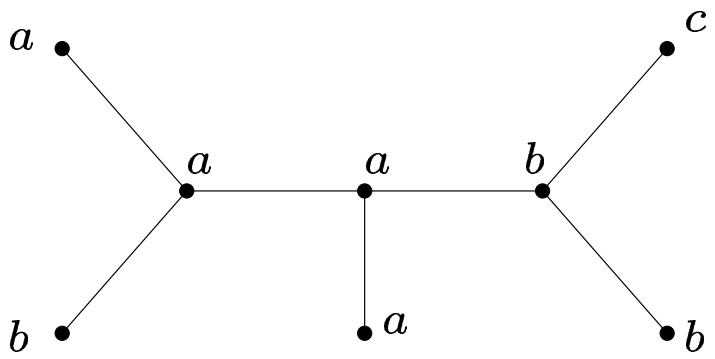
(a)



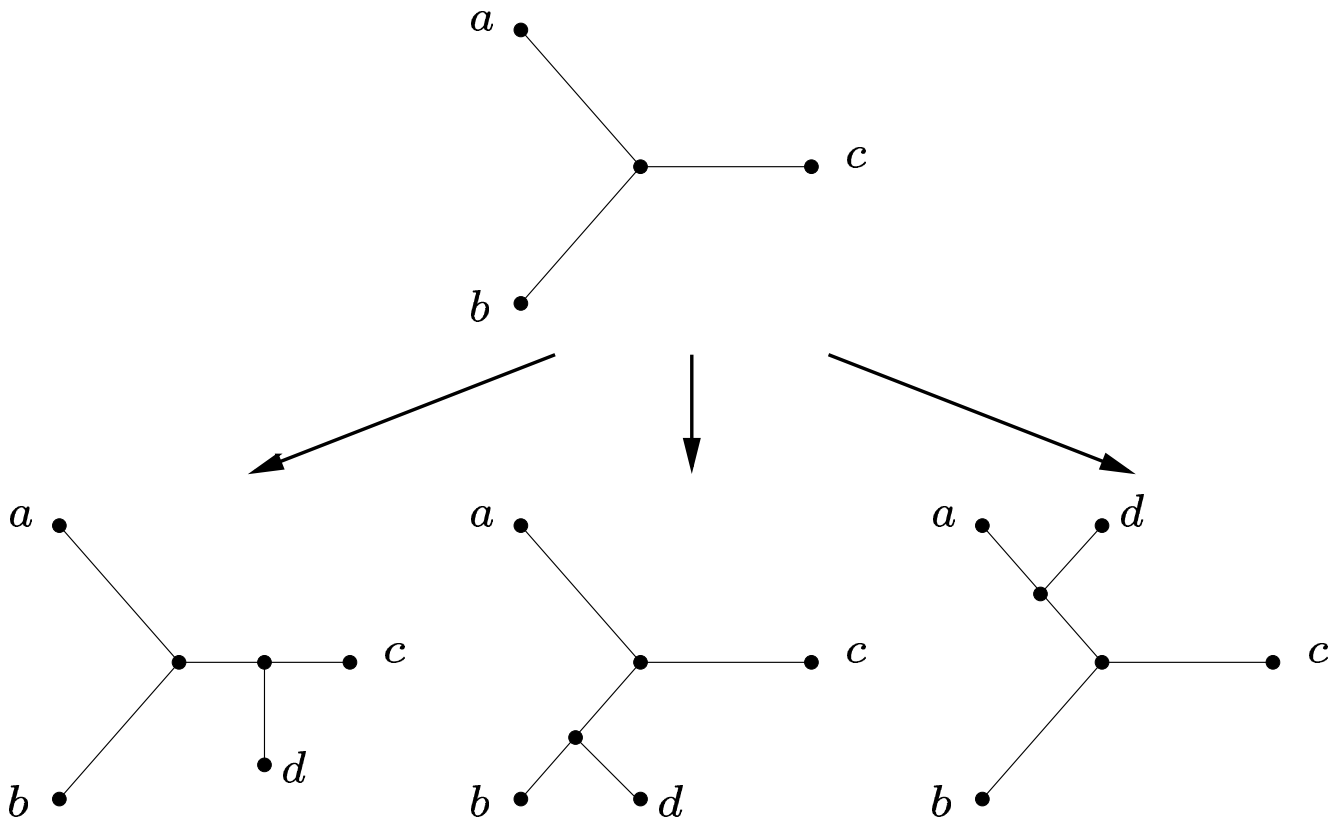
(b)



(c)



(d)


a
c
b
a
c
b
d
a
c
b
d
a
d
c
b

Eingabe: Eine Menge S von n Taxa.

1. Berechne für jede vierelementige Teilmenge $S' \subseteq S$ das optimale Quartett $Q(S')$ (entsprechend der Parsimony-Bewertung).

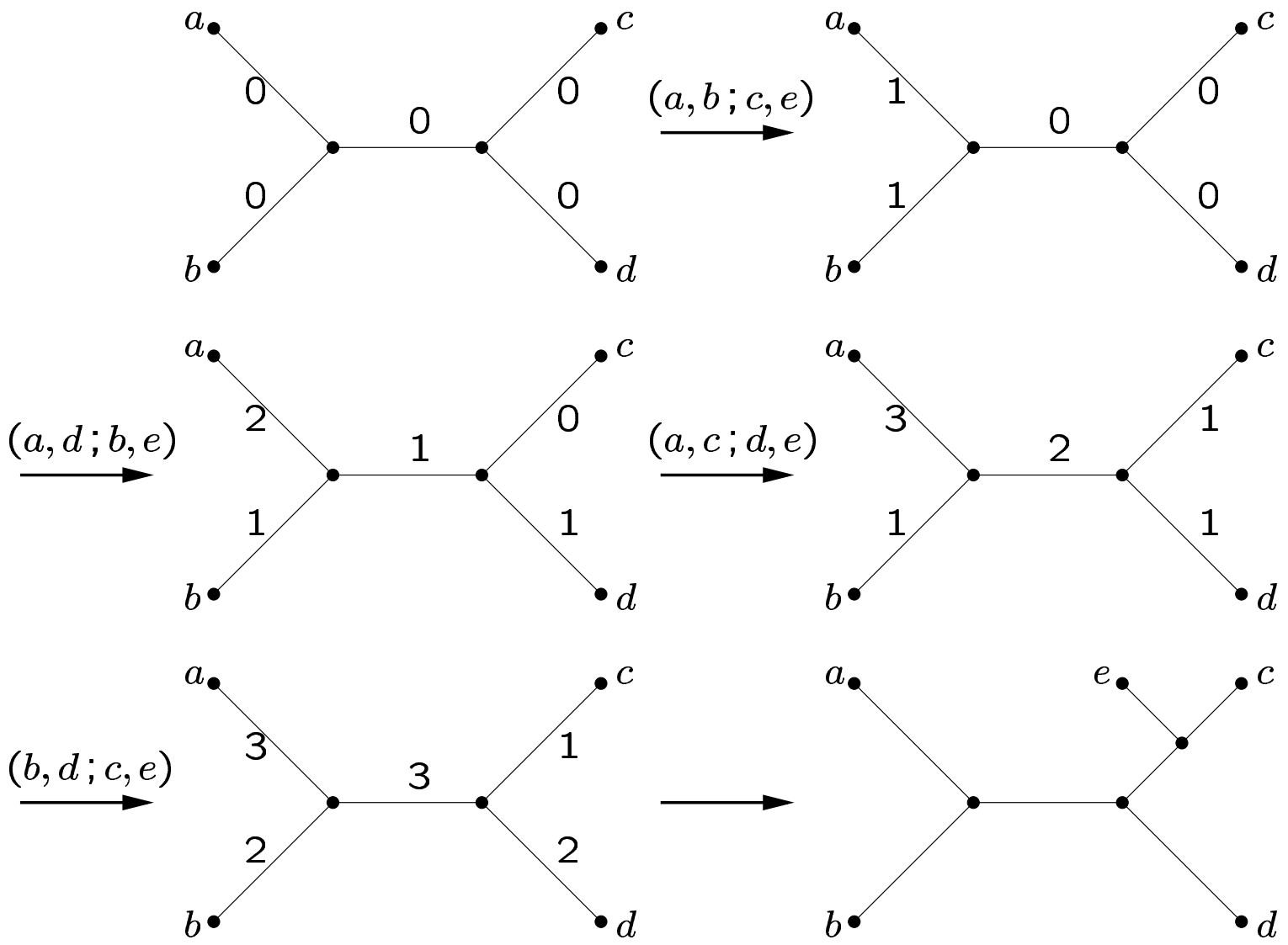
2. Wähle zufällig eine Reihenfolge $a_1, \ldots, a_n$ der Elemente in S.

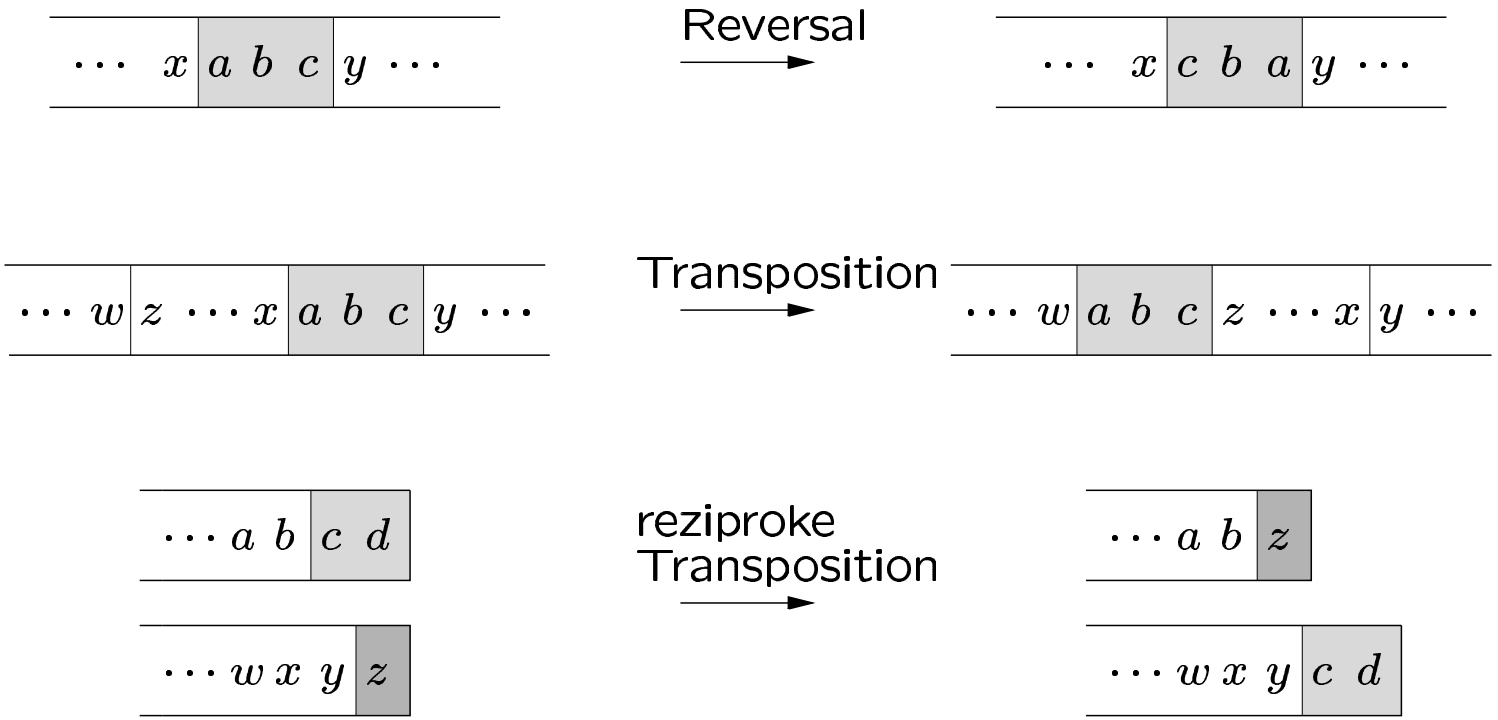
3. Setze $T := Q(\{a_1, a_2, a_3, a_4\})$.

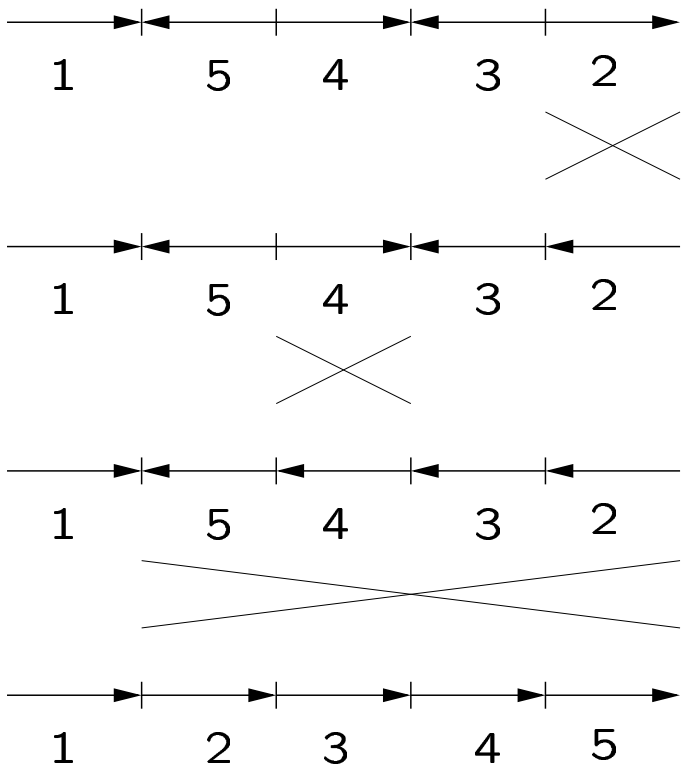
4. **for** $i := 5$ **to** n **do**

   - Initialisiere die Kosten aller Kanten in T mit 0.
   - Für alle $S' = \{b_1, b_2, b_3\} \subseteq \{a_1, \ldots, a_{i-1}\}$, so dass $Q(\{b_1, b_2, b_3, a_i\})$ die Form $(b_1, b_2; b_3, a_i)$ hat, erhöhe die Kantenkosten auf dem Pfad von b_1 nach b_2 in T jeweils um 1.
   - Wähle eine Kante $\{x, y\}$ in T mit minimalen Kosten, lösche diese und füge einen neuen Knoten ein, der mit x, y und dem neuen Blatt a_i verbunden ist.

Ausgabe: Der ungerichtete phylogenetische Baum T für S.


a
b
c
d
e
0
0
0
0
0
$(a,b\,;c,e)$
1
1
0
0
0
$(a,d\,;b,e)$
2
1
1
0
1
$(a,c\,;d,e)$
3
1
2
1
1
$(b,d\,;c,e)$
3
2
3
1
2

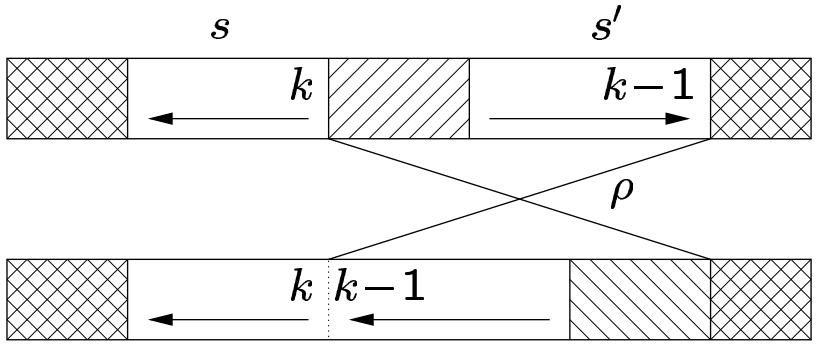

$\cdots\ x\ a\ b\ c\ y\ \cdots$
Reversal
$\cdots\ x\ c\ b\ a\ y\ \cdots$
$\cdots\ w\ z\ \cdots\ x\ a\ b\ c\ y\ \cdots$
Transposition
$\cdots\ w\ a\ b\ c\ z\ \cdots\ x\ y\ \cdots$
$\cdots\ a\ b\ c\ d$
$\cdots\ w\ x\ y\ z$
reziproke
Transposition
$\cdots\ a\ b\ z$
$\cdots\ w\ x\ y\ c\ d$



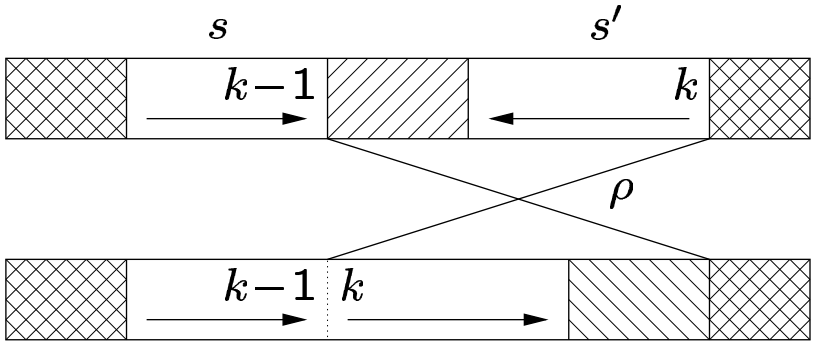
B. oleracea (Kohl)

B. campestris (Rübe)

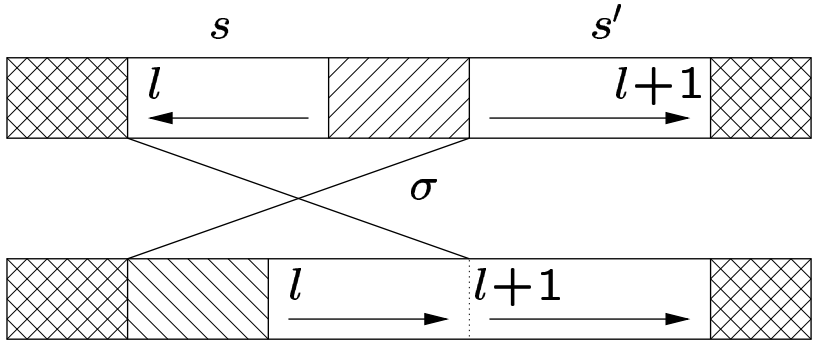
2	1	3	7	5	4	8	6
1	2	3	7	5	4	8	6
1	2	3	4	5	7	8	6
1	2	3	4	5	7	6	8
1	2	3	4	5	6	7	8



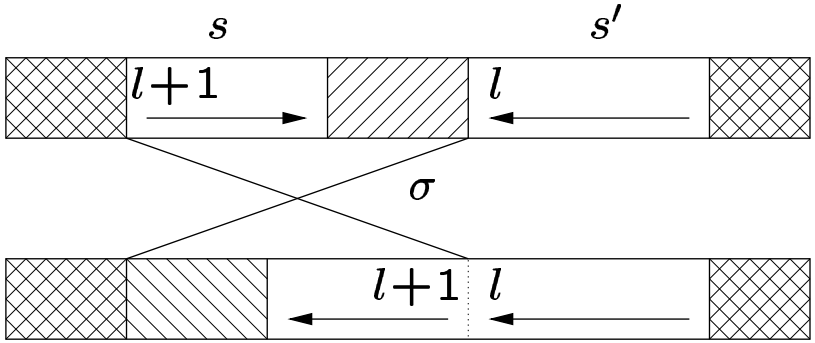
(a)



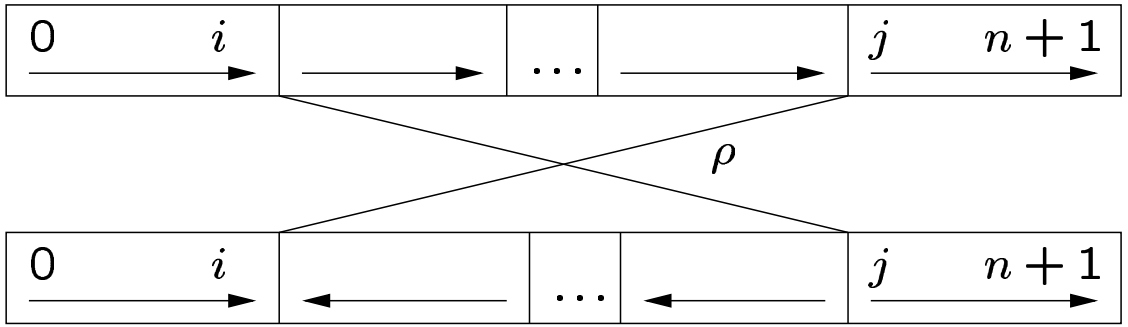
(b)



(c)



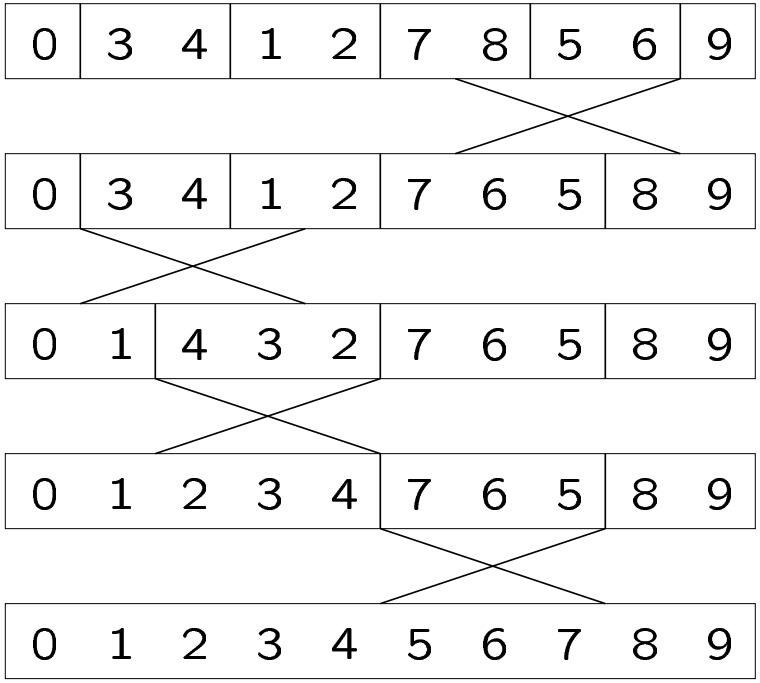
(d)


0
i
$\cdots$
j
$n+1$
ρ
0
i
$\cdots$
j
$n+1$

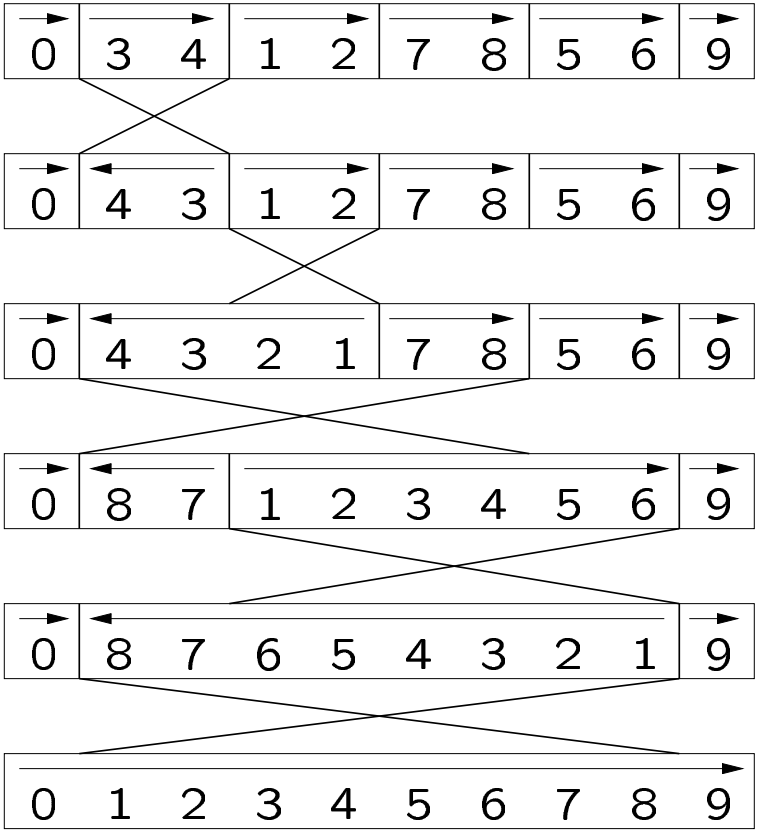
Eingabe: Eine Permutation π der Ordnung n.

$list := \emptyset$
while π ist nicht die identische Permutation **do**
  if π hat absteigenden Strip **then**
    Bestimme das kleinste Element k in einem absteigenden Strip von π.
    Bestimme die Position i von k in π und die Position i' von $k-1$ in π.
    $\rho := (i'+1, i)$-Reversal von $ext(\pi)$
    if $\pi\rho$ hat keinen absteigenden Strip **then**
      Bestimme das größte Element l in einem absteigenden Strip von π.
      Bestimme die Position j von l in π und die Position j' von $l+1$ in π.
      $\rho := (j, j'-1)$-Reversal von $ext(\pi)$
  else $\{\pi$ hat keinen absteigenden Strip$\}$
    $\rho :=$ Reversal, das an den ersten zwei Breakpoints von $ext(\pi)$ schneidet
  $\pi := \pi\rho$
  $list := list \cup \rho$

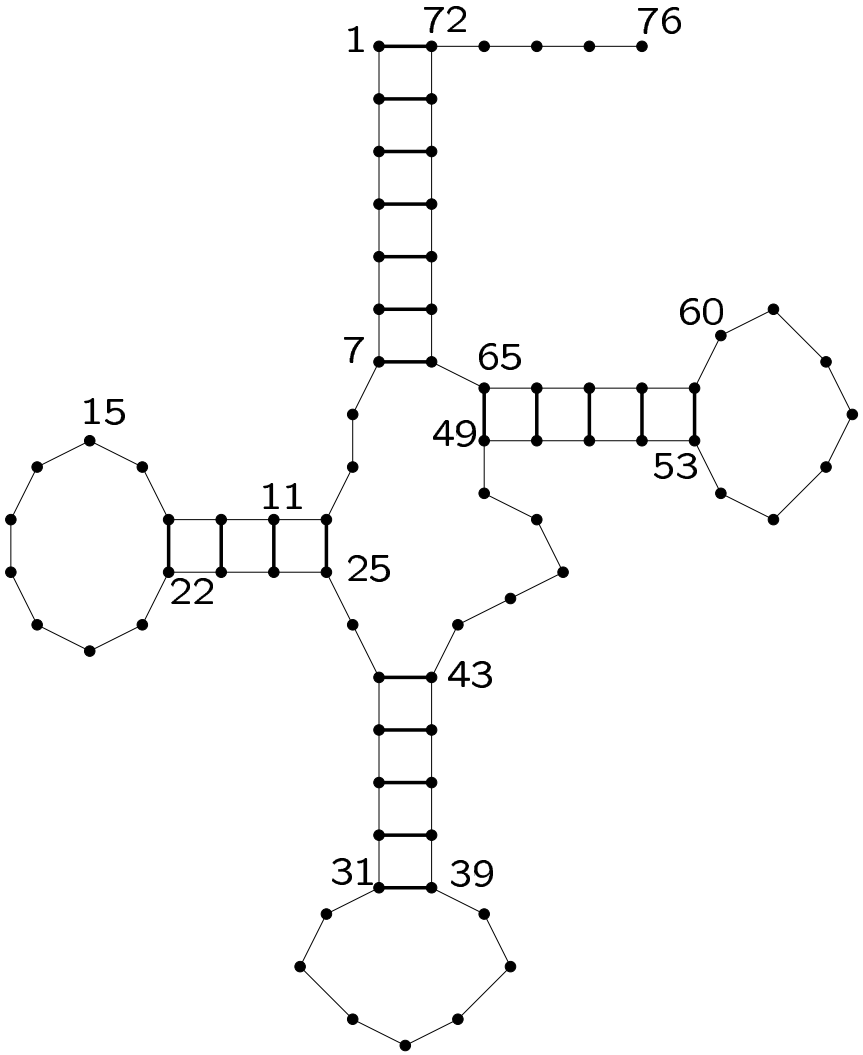
Ausgabe: Die Liste $list$ der Reversals.

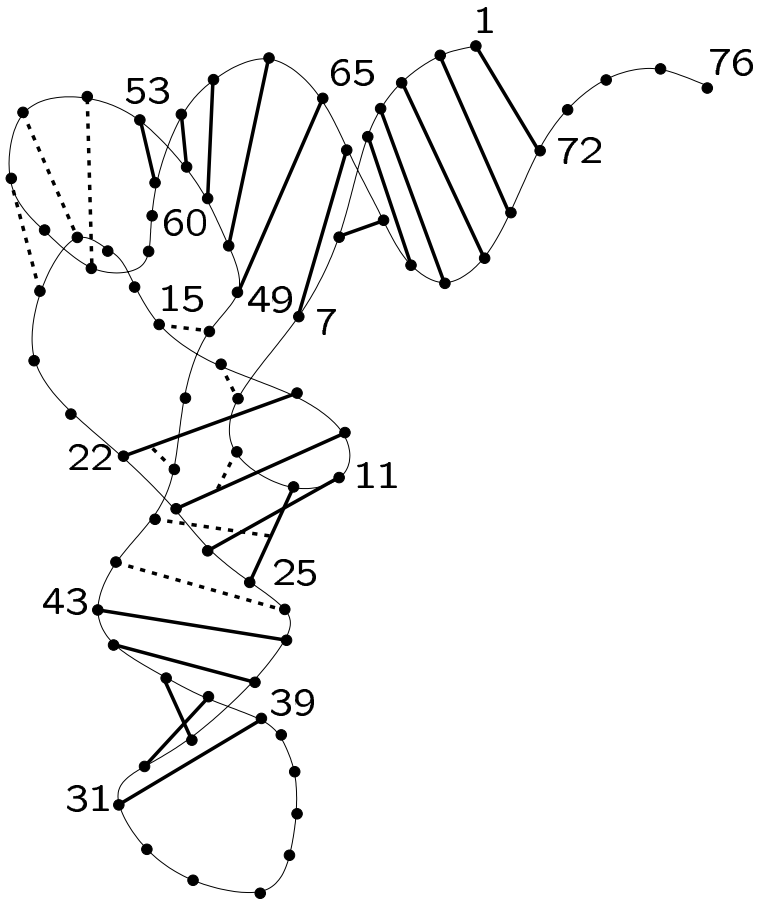

0 3 4 1 2 7 8 5 6 9
0 3 4 1 2 7 6 5 8 9
0 1 4 3 2 7 6 5 8 9
0 1 2 3 4 7 6 5 8 9
0 1 2 3 4 5 6 7 8 9

(a)


0 3 4 1 2 7 8 5 6 9
0 4 3 1 2 7 8 5 6 9
0 4 3 2 1 7 8 5 6 9
0 8 7 1 2 3 4 5 6 9
0 8 7 6 5 4 3 2 1 9
0 1 2 3 4 5 6 7 8 9

(b)


1
72
76
7
65
60
49
53
15
11
25
22
43
31
39


1
76
53
65
72
60
15
49
7
22
11
25
43
39
31

Eingabe: Ein String $r = r_1 \dots r_n$.

1. Initialisierung:

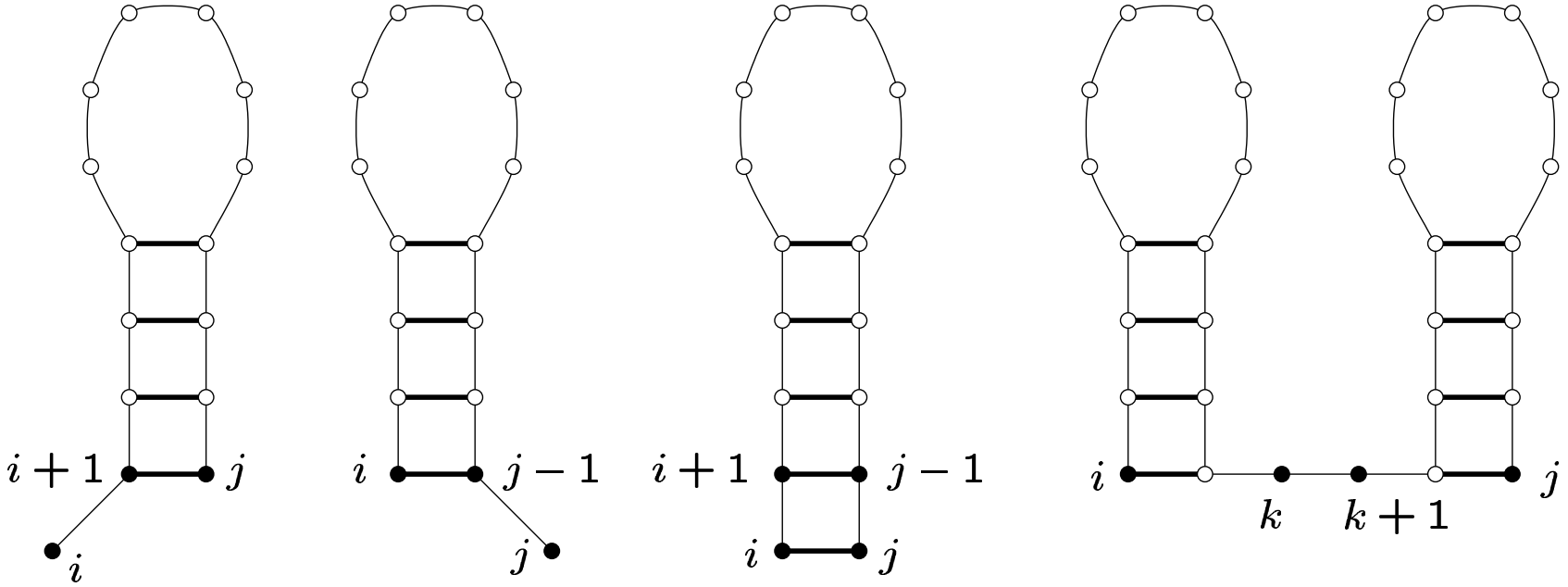
for $i = 2$ to n **do**
  $\mathrm{BP}(S_{i,i-1}) := 0$
for $i = 1$ to n **do**
  $\mathrm{BP}(S_{i,i}) := 0$

2. Rekurrenz:

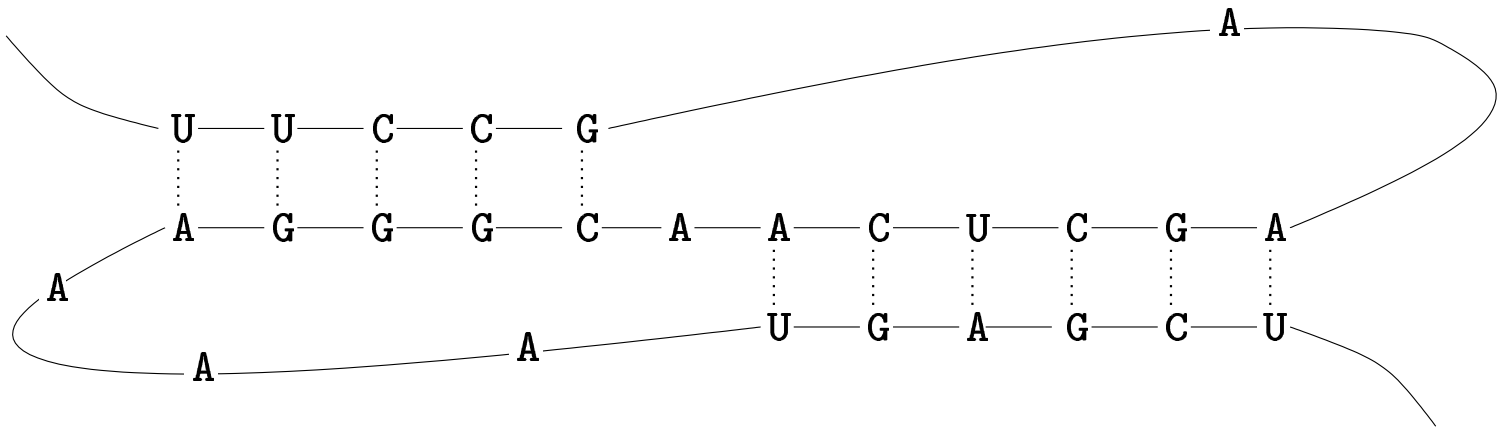
for $l = 1$ to $n-1$ **do**
  for $i = 1$ to $n-l$ **do**
    $j := i + l$;

$$\mathrm{BP}(S_{i,j}) := \max \begin{cases} \mathrm{BP}(S_{i+1,j}) & \text{(i)} \\ \mathrm{BP}(S_{i,j-1}) & \text{(ii)} \\ \mathrm{BP}(S_{i+1,j-1}) + \delta(r_i, r_j) & \text{(iii)} \\ \max_{k, i<k<j}\{\mathrm{BP}(S_{i,k}) + \mathrm{BP}(S_{k+1,j})\} & \text{(iv)} \end{cases}$$

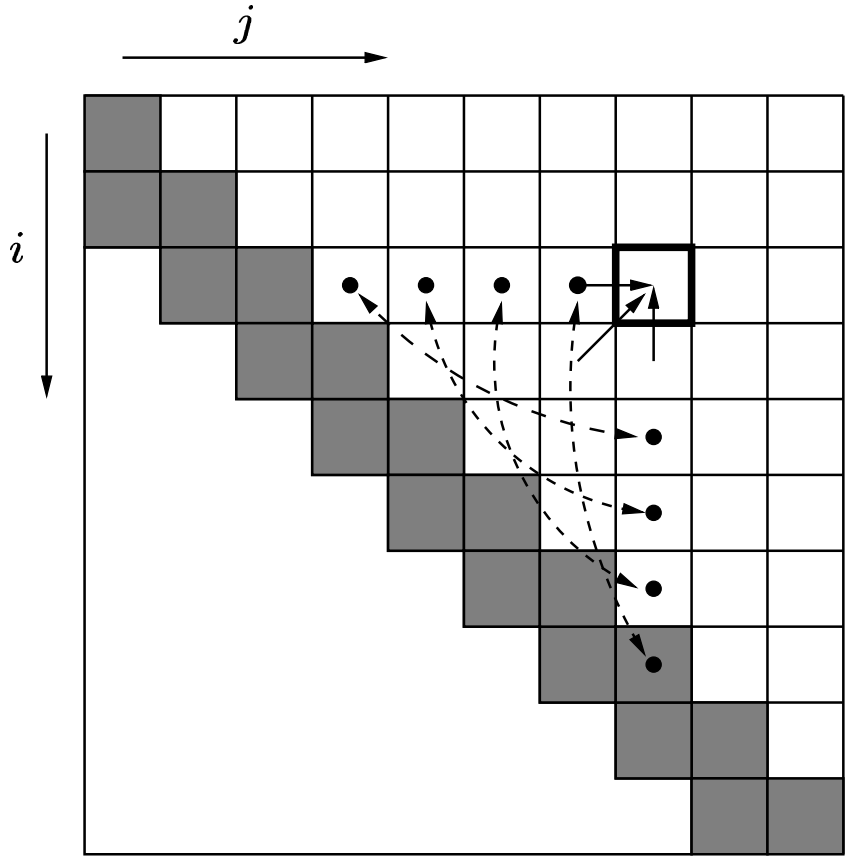
Ausgabe: $\mathrm{BP}(S_{1,n})$

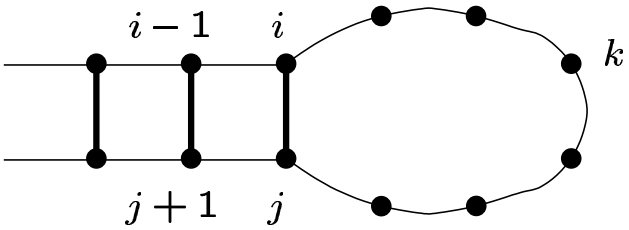


(i) (ii) (iii) (iv)

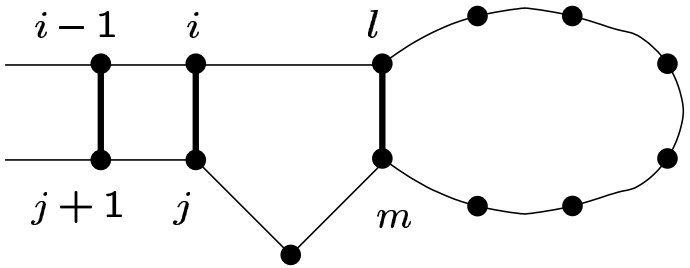

A
U U C C G
A G G G C A A C U C G A
A
U G A G C U
A A

Berechnung der Matrixeinträge BP($S_{i,j}$) (bzw. E($S_{i,j}$)) beim Algorithmus von Nussinov

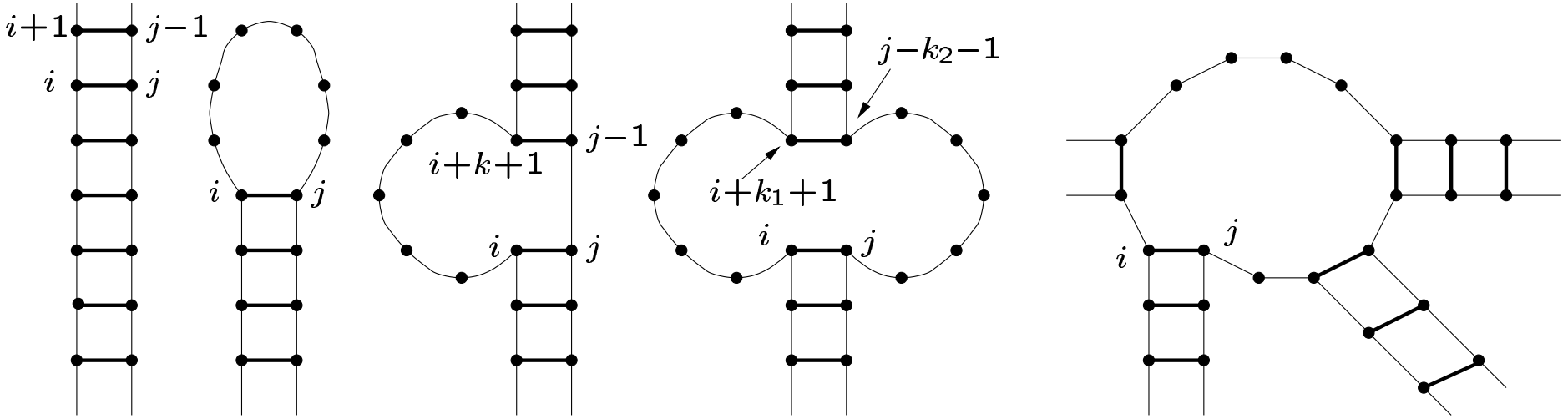




(a)



(b)



Stem

Hairpin Loop

Bulge bei r_i

Interior Loop

Multiple Loop

Eingabe: Ein String $r = r_1 \dots r_n$.

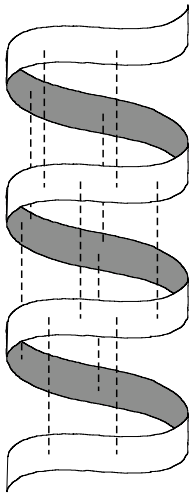
Rekurrenz:

for $l = 1$ to $n - 1$ **do**
  for $i = 1$ to $n - l$ **do**
    $j := i + l$;

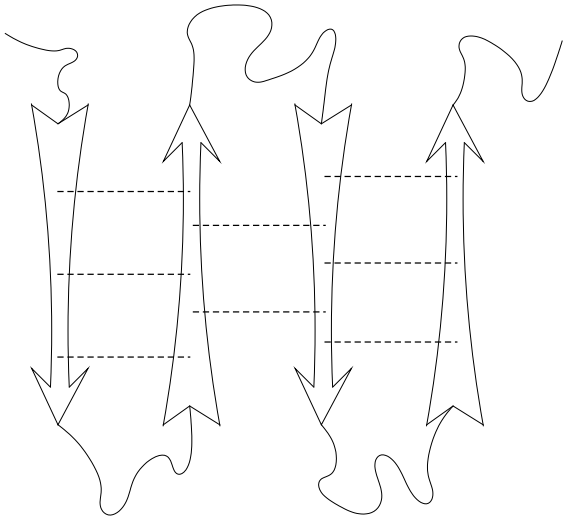
$$\mathsf{E}(S_{i,j}) := \min \begin{cases} \mathsf{E}(S_{i+1,j}) & \text{(i)} \\ \mathsf{E}(S_{i,j-1}) & \text{(ii)} \\ \mathsf{E}(\mathsf{L}_{i,j}) & \text{(iii)} \\ \max_{k,i<k<j}\{\mathsf{E}(S_{i,k}) + \mathsf{E}(S_{k+1,j})\} & \text{(iv)} \end{cases}$$

Ausgabe: $\mathsf{E}(S_{1,n})$

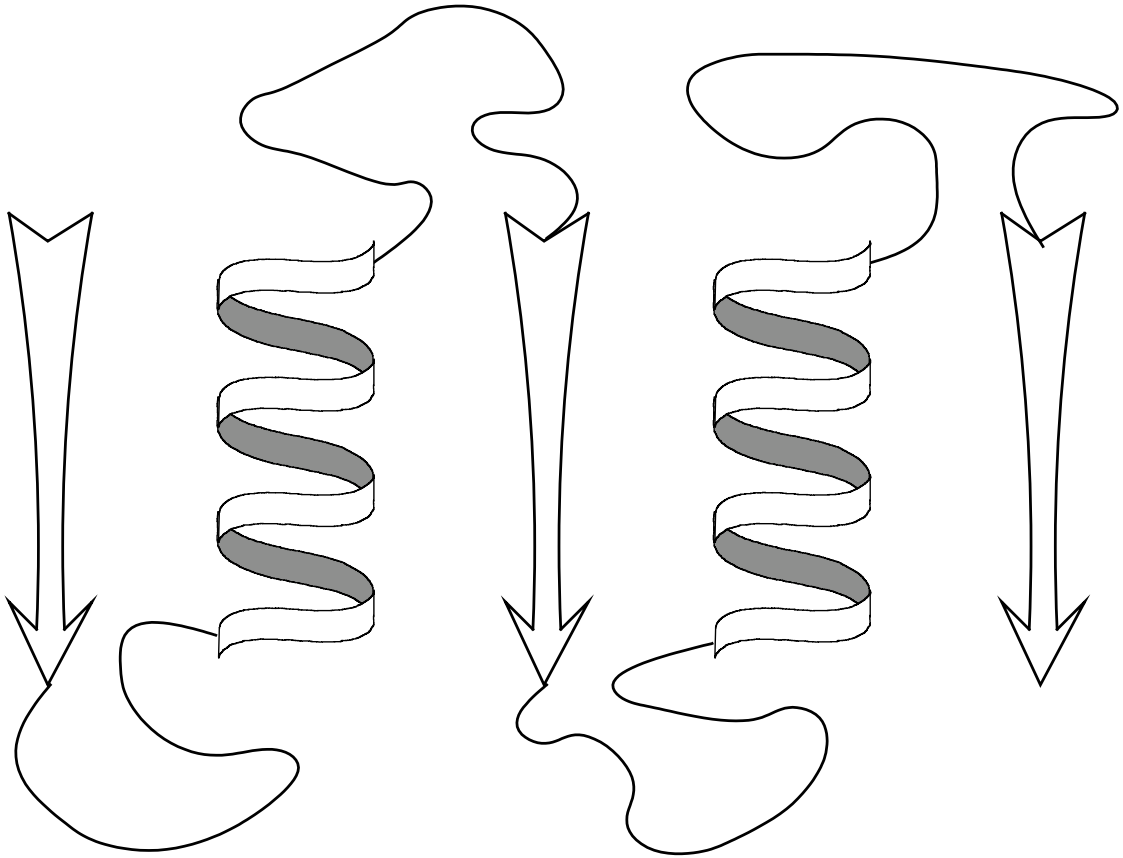
$$E(L_{i,j}) := \begin{cases} fe(r_i, r_j) + fe_{\text{stacked}} + E(S_{i+1,j-1}) \\ \qquad \text{falls } L_{i,j} \text{ ein Stem ist} & \text{(a)} \\ fe(r_i, r_j) + fe_{\text{hairpin}}(j-i-1) \\ \qquad \text{falls } L_{i,j} \text{ ein Hairpin Loop ist} & \text{(b)} \\ \min_{k \geq 1}\{fe(r_i, r_j) + fe_{\text{bulge}}(k) + E(S_{i+k+1,j-1})\} \\ \qquad \text{falls } L_{i,j} \text{ ein Bulge bei } r_i \text{ ist} & \text{(c)} \\ \min_{k \geq 1}\{fe(r_i, r_j) + fe_{\text{bulge}}(k) + E(S_{i+1,j-k-1})\} \\ \qquad \text{falls } L_{i,j} \text{ ein Bulge bei } r_j \text{ ist} & \text{(d)} \\ \min_{k_1,k_2 \geq 1}\{fe(r_i, r_j) + fe_{\text{interior}}(k_1 + k_2) + E(S_{i+k_1+1,j-k_2-1})\} \\ \qquad \text{falls } L_{i,j} \text{ ein Interior Loop ist} & \text{(e)} \end{cases}$$

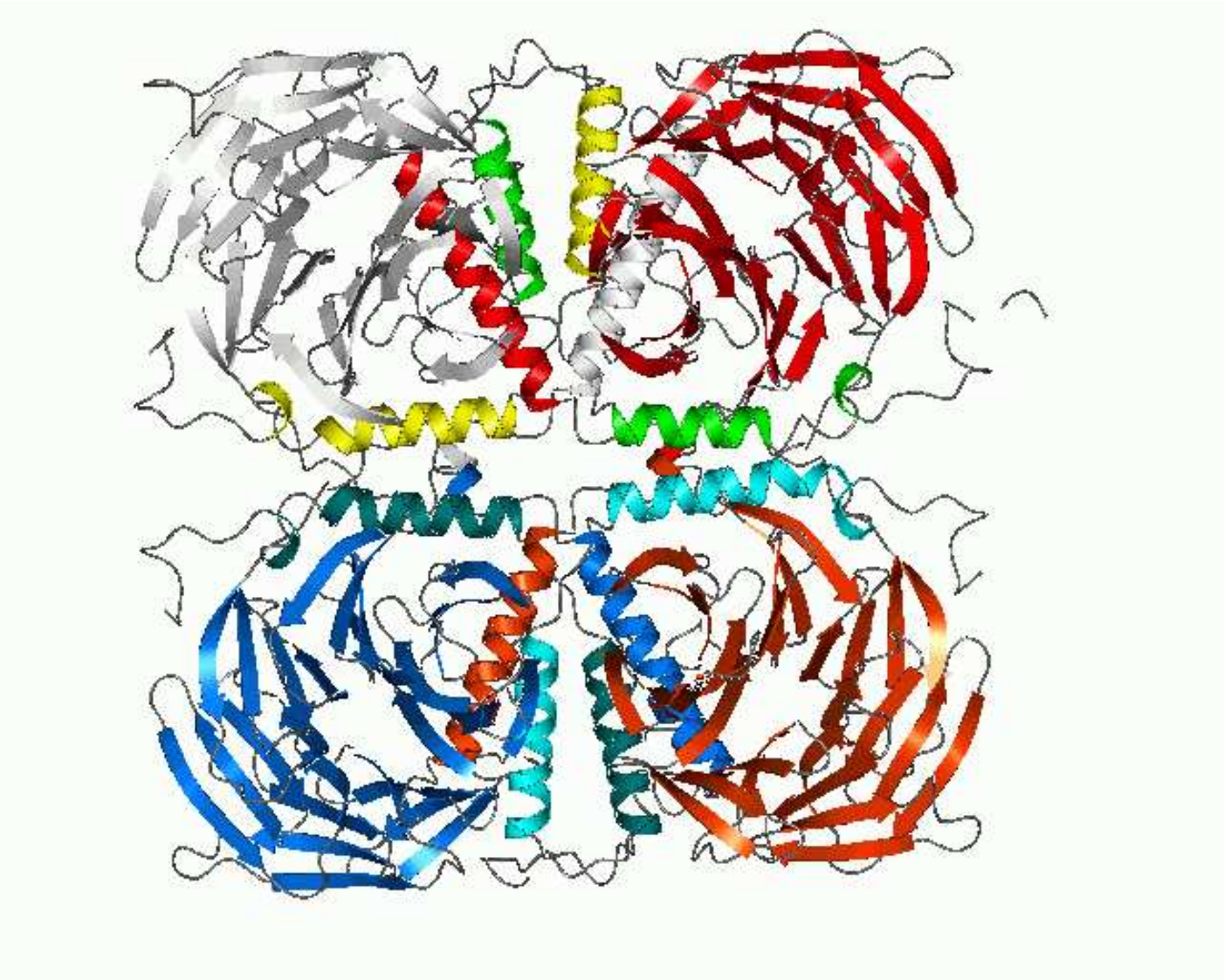


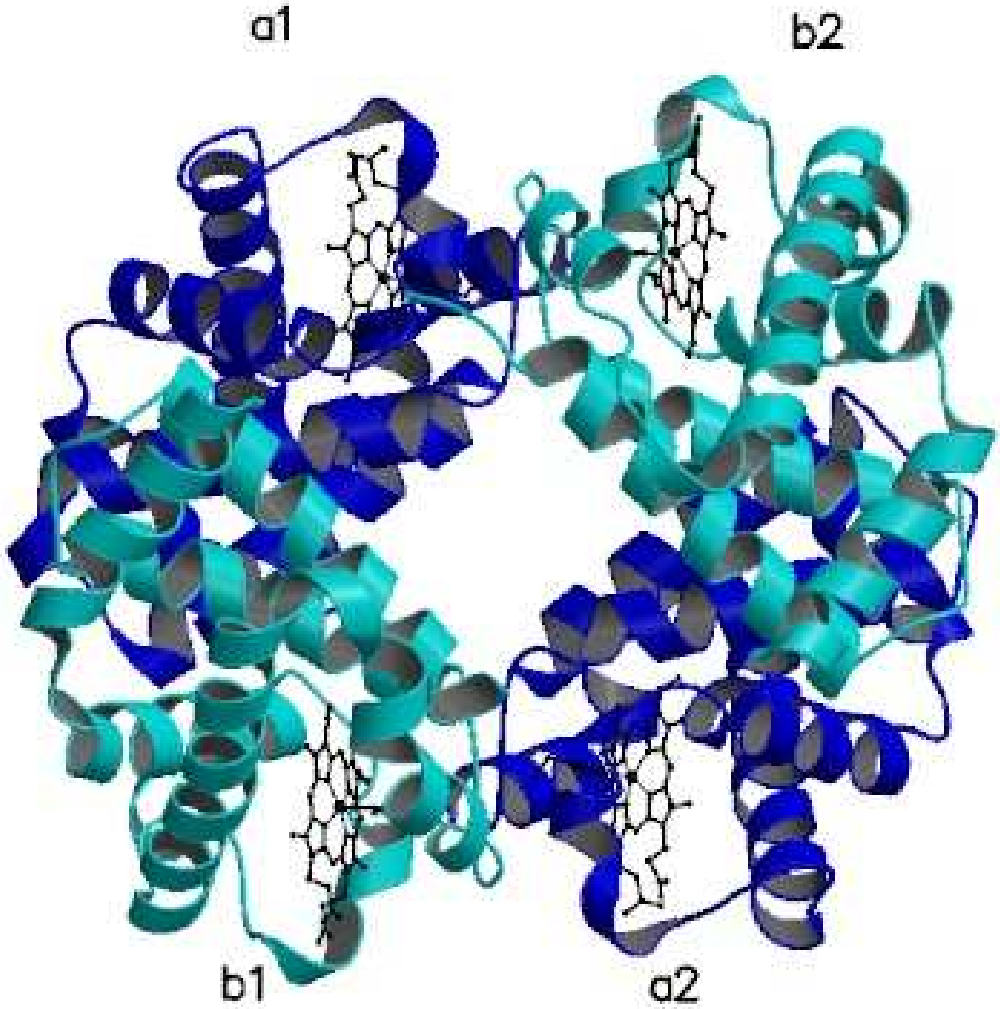
Helix

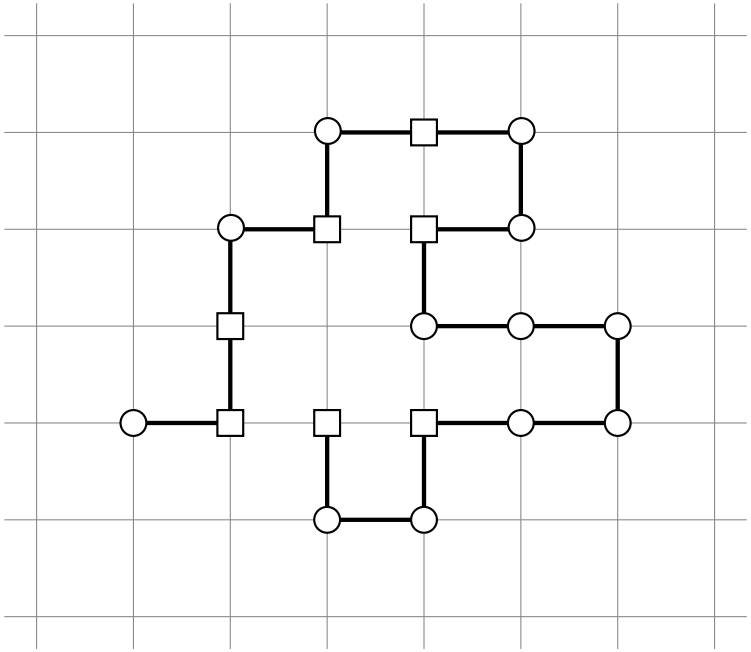


Faltblatt

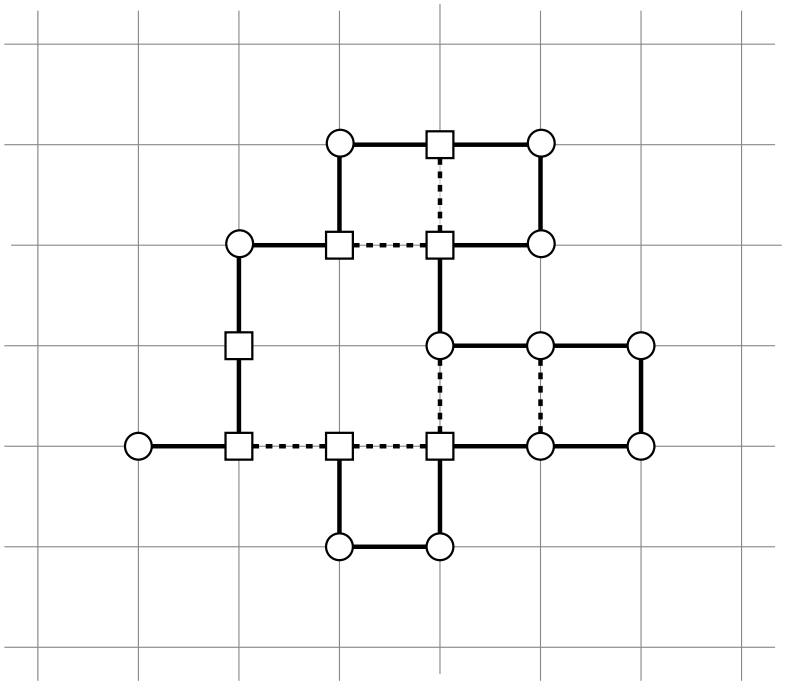




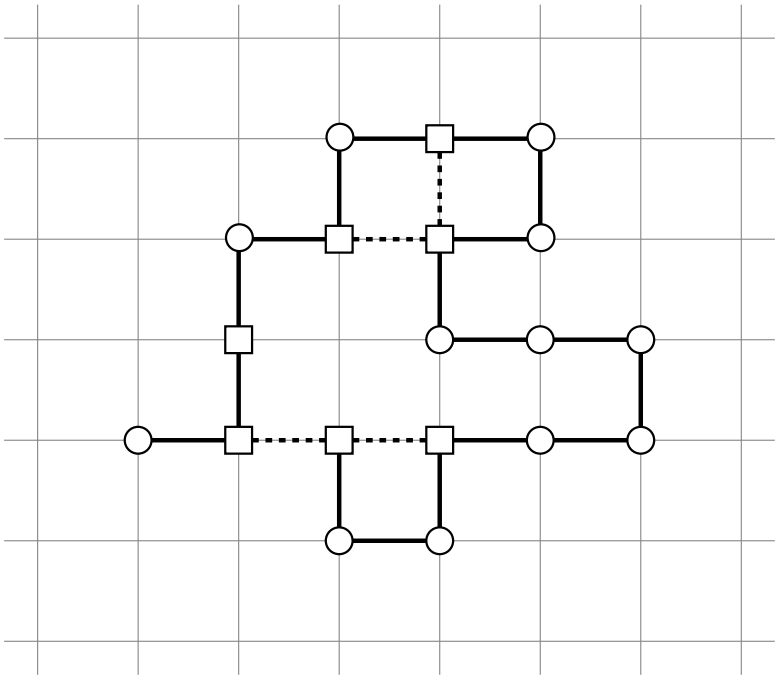

a1
b2
b1
a2



$s = 0110101001000001001$



topologische Nachbarschaften der Einbettung



1-1-Paare der Einbettung

String:

$$s = 0100100110110100110000111011001011 10$$

gerade Eins-Positionen in s: (Anzahl: 8)

$$0\boxed{1}00100\boxed{1}101\boxed{1}0\boxed{1}001\boxed{1}00001\boxed{1}101\boxed{1}00101\boxed{1}10$$

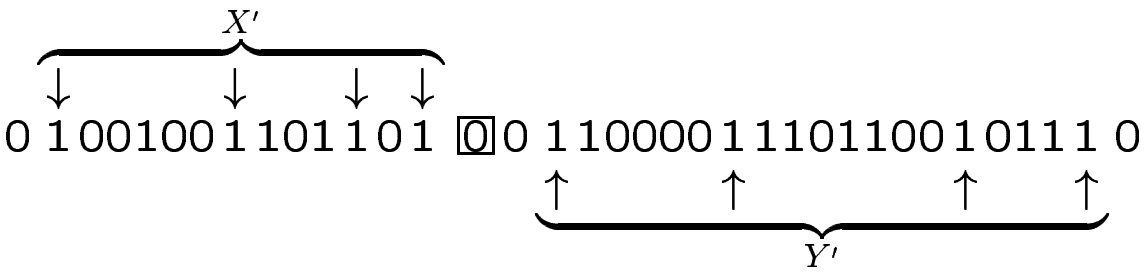
ungerade Eins-Positionen in s: (Anzahl: 10)

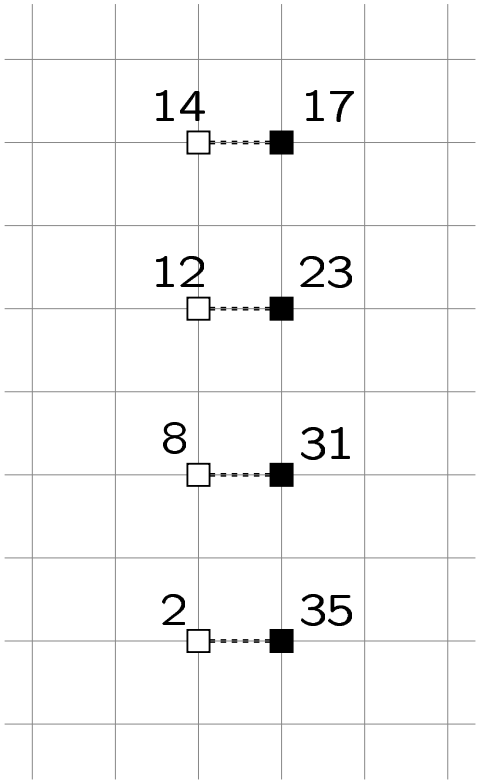
$$0100\boxed{1}001\boxed{1}0\boxed{1}10100\boxed{1}10000\boxed{1}1\boxed{1}0\boxed{1}100\boxed{1}0\boxed{1}1\boxed{1}0$$

XY-Partitionierung von s:

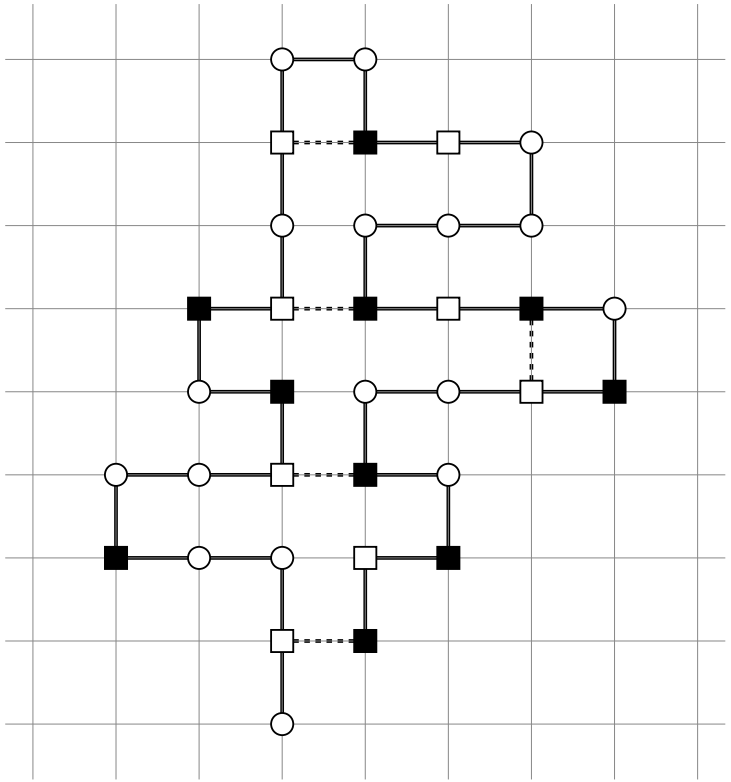
$$\begin{aligned} X &= \{2, 8, 12, 14, 18, 24, 28, 34\} \\ Y &= \{5, 9, 11, 17, 23, 25, 27, 31, 33, 35\} \end{aligned}$$

möglicher Faltungspunkt: $fp = 15 \Rightarrow X' = \{2, 8, 12, 14\}$ und $Y' = \{17, 23, 31, 35\}$





(a)

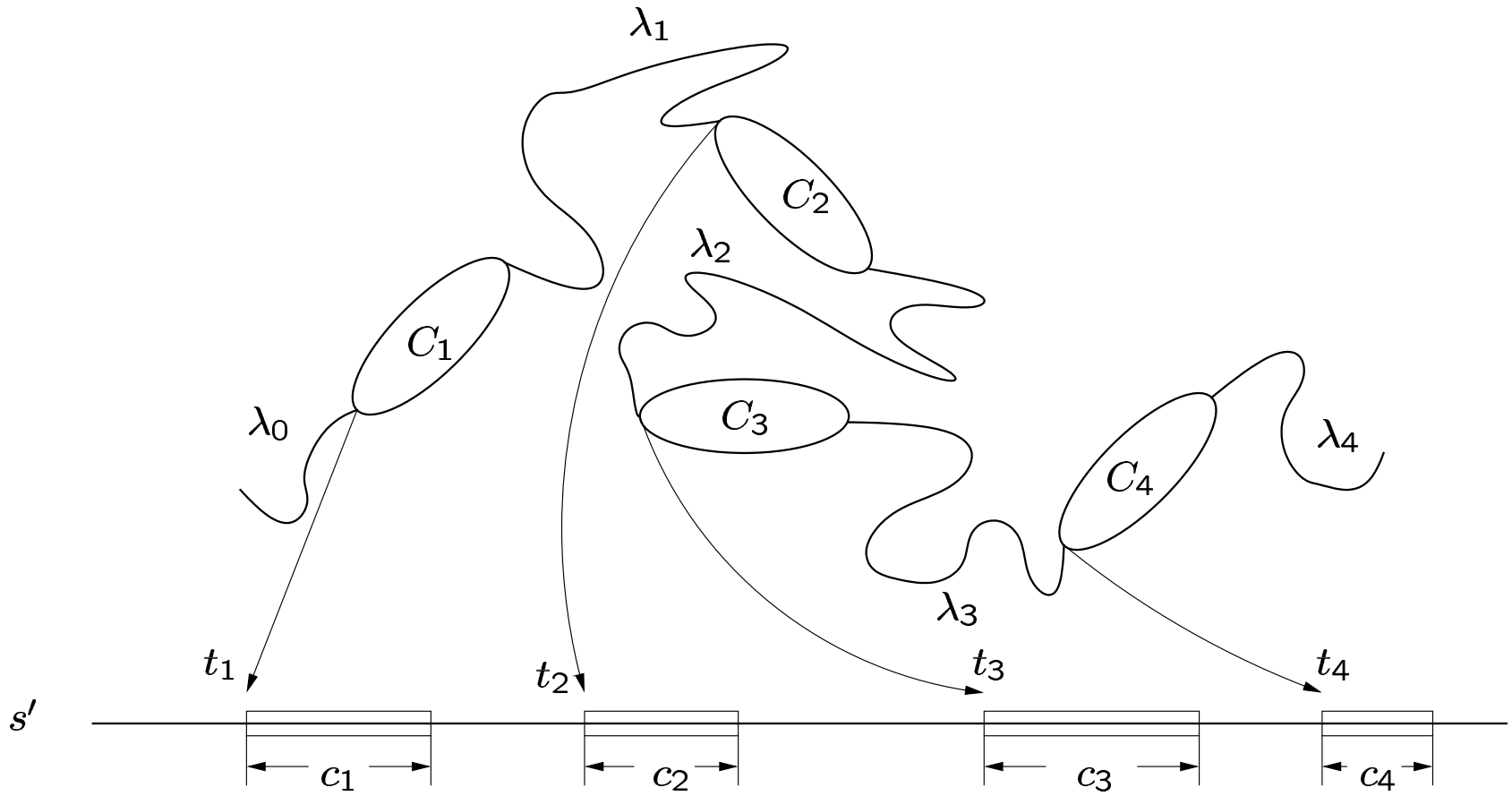


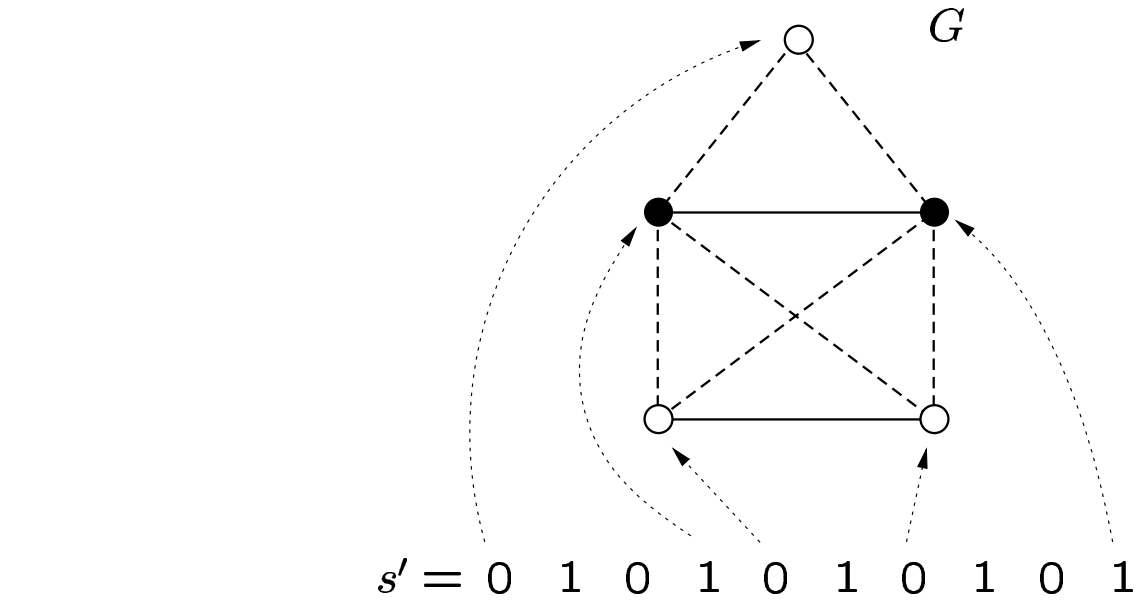
(b)

Eingabe: Ein String $s = s_1 \dots s_n$ über $\{0, 1\}$.

1. Bestimme einen Faltungspunkt fp von s.
2. Bestimme die Positionsmengen X' und Y' mit
   - $X' \subseteq X$ und $|X'| \geq \frac{|X|}{2}$
   - $Y' \subseteq Y$ und $|Y'| \geq \frac{|X|}{2}$
   - X' und Y' liegen auf unterschiedlichen Seiten des Faltungspunktes fp.
3. Ordne die Positionen von X' und Y' so in zwei benachbarten Spalten des Gitters an, dass jede Position in X' Teil eines 1-1-Paares mit einer Position aus Y' ist.
4. Verbinde die bisher festgelegten Positionen in Form U-förmiger Seitenarme.

Ausgabe: Die durch die obigen Schritte induzierte Einbettung ξ von s in $\mathbb{Z}^2$.


λ_1
C_2
λ_2
C_1
λ_0
C_3
λ_4
C_4
λ_3
t_1
t_2
t_3
t_4
s'
c_1
c_2
c_3
c_4


G
$s' = 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1$

Eingabe: Ein Strukturmodell $M = (m, c, \lambda, l_{\text{min}}, l_{\text{max}})$ eines Strings s, ein String s', zwei Funktionen g_1 und g_2, und eine Untere-Schranken-Funktion lowbound.

1. Initialisierung:

   $opt := \infty$ {Kosten des aktuell besten Threadings}
   $thr :=$ nicht definiert {Aktuell bestes Threading}
   $\mathcal{T} := ([0,\infty], \ldots, [0,\infty])$ {Menge aller Threadings}
   $lb :=$ lowbound$(\mathcal{T})$
   $Q :=$ Enqueue$(Q, (\mathcal{T}, lb))$

2. Branch and Bound:

   while $Q \neq$ NIL **do**
   $(\mathcal{T}_i, lb_i) :=$ Dequeue(Q)
   if $lb_i < opt$ **then**
   if $|\mathcal{T}_i| = 1$ **then**
   sei t das einzige gültige Threading in $\mathcal{T}_i$
   if $cost(t) < opt$ **then**
   $opt := cost(t)$; $thr := t$
   else
   Wähle j und d_j mit $1 \leq j \leq m$ und $b_j < d_j < e_j$ für $\mathcal{T}_i$.
   $Q :=$ Enqueue$(Q, (\mathcal{T}_i[b_j, d_j - 1],$ lowbound$(\mathcal{T}_i[b_j, d_j - 1])))$
   $Q :=$ Enqueue$(Q, (\mathcal{T}_i[d_j, d_j],$ lowbound$(\mathcal{T}_i[d_j, d_j])))$
   $Q :=$ Enqueue$(Q, (\mathcal{T}_i[d_j + 1, e_j],$ lowbound$(\mathcal{T}_i[d_j + 1, e_j])))$

Ausgabe: Ein optimales Threading thr mit Kosten opt.